SHEAR-WAVE SEISMIC REFLECTION EXPLORATION FOR CAVITIES AND TUNNELS

VOLUME II – APPENDICES

FINAL TECHNICAL REPORT

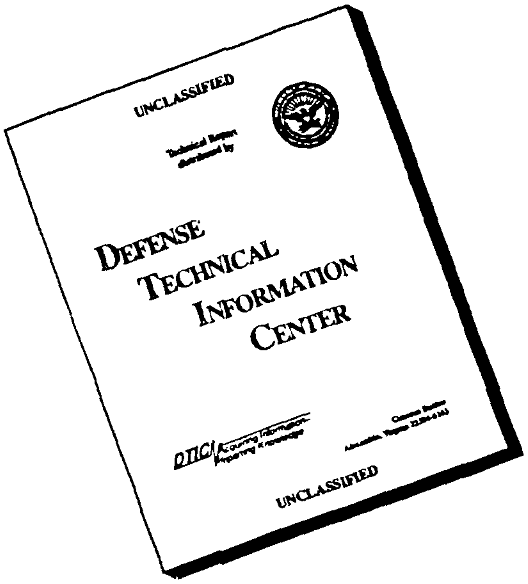
Contract No. DACA39-86-K-0017
SwRI Project No. 14-1421

Prepared for:
U.S. Army Engineer Waterways Experiment Station
Corps of Engineers
P.O. Box 631
Vicksburg, MS 39180-0631

September 1987

SOUTHWEST RESEARCH INSTITUTE
SAN ANTONIO HOUSTON

DISCLAIMER NOTICE


UNCLASSIFIED
Technical Report distributed by
DEFENSE TECHNICAL INFORMATION CENTER
DTIC Acquiring Information Imparting Knowledge
UNCLASSIFIED

United States Government

ATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704 0188
Exp Date Jun 30 1986

TY CLASSIFICATION
ified

1b RESTRICTIVE MARKINGS

ASSIFICATION AUTHORITY

3 DISTRIBUTION/AVAILABILITY OF REPORT
Approved for public release;
Distribution unlimited

ATION/DOWNGRADING SCHEDULE

ORGANIZATION REPORT NUMBER(S)

5 MONITORING ORGANIZATION REPORT NUMBER(S)

ORGANIZATION
Institute

6b OFFICE SYMBOL
(If applicable)

7a NAME OF MONITORING ORGANIZATION
USAEWES Geotechnical Laboratory

, 6220 Culebra Road
, TX 78284

7b ADDRESS *(City, State, and ZIP Code)*
P.O. Box 631
Vicksburg, MS 39180-0631

FUNDING/SPONSORING
ATION
y Belvoir Research,
nt, & Engineering Center

8b OFFICE SYMBOL
(If applicable)

9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
DACA39-86-K-0017

(City, State, and ZIP Code)
Belvoir, VA 22060-5606

10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO

Security Classification)
Wave Seismic Reflection Exploration for Cavities and Tunnels,
and II.

AUTHOR(S)
, Thomas E., Parra, Jorge O., and Biard, James C.

OF REPORT
al Report

13b TIME COVERED
FROM ________ TO ________

14 DATE OF REPORT *(Year, Month, Day)*
January 1988

15 PAGE COUNT

SUPPLEMENTARY NOTATION

COSATI CODES		
FIELD	GROUP	SUB-GROUP

18 SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*

Tunnels
Tunnel Detection
Seismic
Reflection
Polarized Waves
Cavities

ABSTRACT *(Continue on reverse if necessary and identify by block number)*

A theoretical analysis of horizontally polarized shear (SH) waves was performed
determine the seismic reflection characteristics and practical seismic frequency
ge for detecting cylindrical cavities representative of man-made tunnels. The
dimensional analytical model consists of an SH-wave line source in an absorptive
-layered half-space containing a cylindrical cavity in a high-quality bedrock
r underlying a lossy low-velocity surface layer. Seismic wave scattering patterns
synthetic reflection seismograms were computed to illustrate the reflection
ponse of SH waves polarized parallel to the cavity axis. The influence of the
sy surface layer on the reflected SH waves was found to be significant enough
require the use of subsurface-coupled source and detectors for practical detection
the cavity. Based upon these analytical model results, technical specifications
e developed for a high-resolution SH-wave seismic exploration system designed
cifically for search and detection of tunnels. A borehole-coupled asymmetrical (cont'd)

STRIBUTION/AVAILABILITY OF ABSTRACT
UNCLASSIFIED/UNLIMITED □ SAME AS RPT □ DTIC USERS

21 ABSTRACT SECURITY CLASSIFICATION
Unclassified

NAME OF RESPONSIBLE INDIVIDUAL

22b TELEPHONE *(Include Area Code)*

22c OFFICE SYMBOL

83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

19. Abstract (cont'd)

force transducer operating on the electrical arc discharge principle is recommended as the source device and three-component borehole-coupled seismic detectors are recommended for complete SH-wave data acquisition. This seismic exploration system, designed to operate in the frequency range of 400-1,600 Hz together with specialized data processing techniques capable of preserving the high-frequency timing accuracies in the seismic signals and enhancing reflections from localized cavity targets, is recommended for future prototype development and field evalaution.

FOREWORD

VOLUME II - APPENDICES

The information presented herein is supplemental to the theoretical and computational analyses concerning shear-wave seismic reflection exploration for cavities and tunnels presented in Volume I - Study and Design of Techniques.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (cont'd)

LIST OF ILLUSTRATIONS (cont'd)

APPENDIX A

THE GREEN'S FUNCTION FOR THE UNBOUNDED MEDIUM

APPENDIX A

The Green's Function for the Unbounded Medium

The two-dimensional Green's function for an unbounded medium in the absence of a surface layer is a solution of the inhomogeneous wave equation

$$(\nabla^2+k^2)G(x,y;x',y') = -\delta(x-x')\delta(y-y') \ . \qquad \text{(A-1)}$$

To find the solution, the wave equation is first referred to a cartesian coordinate system (x,y) having an SH-wave line source at the origin

$$(\nabla^2+k^2)G(x,y) = -\delta(x)\delta(y) \ , \qquad \text{(A-2)}$$

and is then transformed to a wave equation in polar coordinates. Thus, the identity

$$\int\delta(r)dr = 1 = \iint_S \delta(x)\delta(y)dxdy = \iint_S rdrd\phi\delta(x)\delta(y) \ ,$$

leads to

$$\delta(x)\delta(y) = \frac{\delta(r)}{2\pi r}$$

and the polar coordinate form of the wave equation is

$$\left[\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial}{\partial r}\right) + k^2\right]G(r) = -\frac{\delta(r)}{2\pi r} \ . \qquad \text{(A-3)}$$

For r ≠ 0, this equation becomes

$$\left[\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial}{\partial r}\right) + k^2\right]G(r) = 0 \ . \qquad \text{(A-4)}$$

The solution of this equation is the Hankel function of zeroth order. Since Equation (A-4) exhibits a singularity at r = 0 and the Hankel function solution should represent an outgoing wave, then

$$G(r) = cH_0^{(1)}(kr) \ . \qquad \text{(A-5)}$$

The constant c may be determined by matching the boundary condition at r = 0. Thus, integrating Equation (A-3) over an infinitesimal area 2πrdr and letting r → 0, results in

$$\lim_{r \to 0} \left\{ 2\pi r\left(\frac{\partial G(r)}{\partial r}\right) + k^2 \int 2\pi r G(r) dr \right\} = -1 \ . \tag{A-6}$$

Using the asymptotic expression for the Hankel function

$$H_o^{(1)}(kr) \underset{kr \to 0}{\to} j \frac{2}{\pi} \log (kr) \ .$$

and substituting $G(r) = c \frac{j2}{\pi} \log (kr)$ into Equation (A-6) yields

$$\lim_{r \to 0} \left\{ 4cj + k^2 4cj \int r \log (kr) dr \right\} = -1$$

from which it follows that

$$c = \frac{j}{4}$$

and, from Equation (A-5),

$$G(r) = \frac{j}{4} H_o^{(1)}(kr) \ .$$

This result is the two-dimensional Green's function in cylindrical coordinates. A more general representation of this solution for an SH-wave line source at (r',ϕ') and a detector point located at (r,ϕ) is

$$G(r,r') = \frac{1}{4\pi} \left[j\pi H_o^{(1)}(k|r-r'|\right] \tag{A-7}$$

Introducing the spectral representation of the Hankel function as given by Stinson (1976), the free-space Green's function in cartesian coordinates takes the form

$$G(x,y;x',y') = \frac{j}{4\pi} \int_{-\infty}^{\infty} \frac{e^{j\alpha(x-x')}}{\alpha} e^{j\lambda(y-y')} d\lambda \tag{A-8}$$

where:

$$\alpha = (k^2 - \lambda^2)^{1/2} .$$

Or,

$$G(x,y;x',y') = \frac{1}{4\pi} \int_{-\infty}^{\infty} \frac{e^{-\gamma(x-x')}}{\gamma} e^{j\lambda(y-y')} d\lambda \ ; \ \text{for } \gamma = (\lambda^2 - k^2)^{1/2} \qquad \text{(A-9)}$$

APPENDIX B

THE INTEGRAL EQUATION FORMULATION

APPENDIX B

The Integral Equation Formulation

The total wave displacement u^t in a medium is composed of two parts: the incident wave $u^{(i)}$ and the scattered wave $u^{(s)}$. Thus,

$$u^t = u^{(i)} + u^{(s)} \ . \tag{B-1}$$

Each wave function satisfies the Helmholtz first (interior) and second (exterior) formulas. The geometry representing the first Helmholtz equation is illustrated in Figure B-1. This first equation is given by

$$\iint_S \left[G(r,r') \frac{\partial u(r')}{\partial \hat{r}} - u(r') \frac{\partial G(r,r')}{\partial \hat{r}} \right] dS = \begin{cases} u(r); \text{ for } r \text{ inside } S \\ 0; \text{ for } r \text{ outside } S \end{cases} \tag{B-2}$$

where G(r,r') is the Green's function.

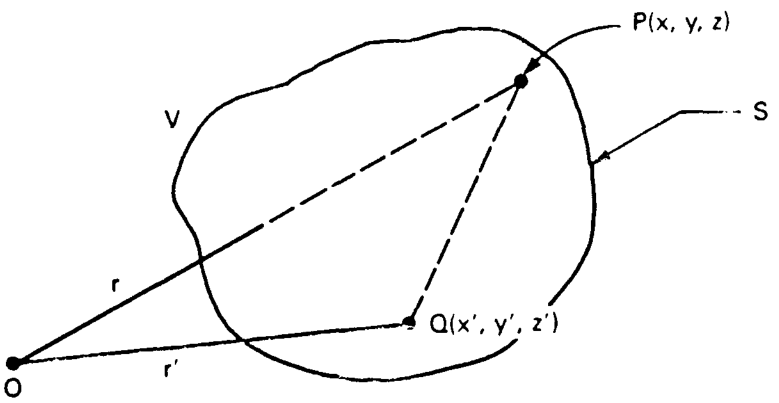


FIGURE B-1. GEOMETRY OF THE OBSERVATION POINT P(r) AND SOURCE POINT Q(r') FOR THE HELMHOLTZ INTERIOR PROBLEM

The Helmholtz interior formula is applicable to the case when all of the singularities of the function u(r) lie outside of the surface S. That is, it is valid only when the function u(r) and its first and second partial derivatives are continuous on or inside of the surface S.

Alternatively, if all of the singularities of u(r) lie within a closed surface, S, the Green's identity given by

$$\iiint_V (u\nabla^2 G - G\nabla^2 u)dV = \iint_S \left(u \frac{\partial G}{\partial \hat{r}} - G \frac{\partial u}{\partial \hat{r}}\right) dS \ , \tag{B-3}$$

may be applied to the region V bounded internally by S and externally by another closed surface Γ. That is, for example, by the surface of a sphere with the center at the origin and having a large radius, R, as illustrated in Figure B-2.

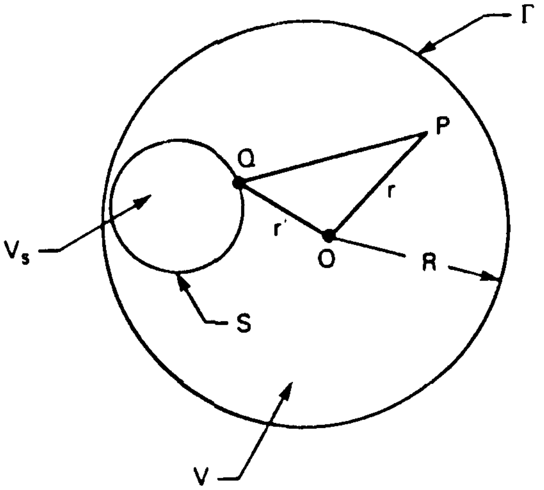


FIGURE B-2. GEOMETRY FOR THE OBSERVATION POINT P(r) AND A SOURCE POINT Q(r') FOR WAVES EXTERIOR TO THE SURFACE S

For this case, the Helmholtz second (exterior) formula is used and is given by

$$\iint_S \left[u(r') \frac{\partial G(r,r')}{\partial \hat{r}'} - G(r,r') \frac{\partial u(r')}{\partial \hat{r}'}\right] dS = \begin{cases} u(r); \text{ for } r \text{ outside } S \\ 0; \text{ for } r \text{ inside } S \ . \end{cases} \tag{B-4}$$

The scattered wave displacement, u^s, which physically represents the waves radiated by secondary sources on or inside the surface, S, usually is singular inside V_S. Thus, Helmholtz's second formula is applicable with

$$\iint_S \left[u^s(r') \frac{\partial G(r,r')}{\partial \hat{r}} - G(r,r') \frac{\partial u^s(r,r')}{\partial \hat{r}}\right] dS = u^s(r) \ ; \text{ for } r \text{ in } V \ . \tag{B-5}$$

In addition, the incident wave displacement, $u^{(i)}$, which has no singularity inside the boundary, S, satisfies the first Helmholtz formula

$$\iint_S \left[G(r,r') \frac{\partial u^{(i)}(r')}{\partial \hat{r}'} - u^{(i)}(r') \frac{\partial G(r,r')}{\partial \hat{r}'}\right] dS = 0 \ ; \text{ for } r \text{ in } V \ . \qquad \text{(B-6)}$$

Adding Equations (B-5) and (B-6) and substituting the result into Equation (B-1) yields the total wave displacement function as

$$u^{(t)} = u^{(i)} + \iint_S \left[u^{(t)}(r') \frac{\partial G(r,r')}{\partial \hat{r}'} - G(r,r') \frac{\partial u^{(t)}(r')}{\partial \hat{r}'}\right] dS \ ; \text{ for } r \text{ in } V \ . \qquad \text{(B-7)}$$

Since the Green's function G(r,r') is known, Equation (B-7) states that the scattered waves in the region V, outside of the boundary surface S of the scatterer, can be found from an integration over S if the values of $u^{(t)}$ and $\frac{\partial u^{(t)}}{\partial \hat{r}'}$ on the surface are known. For example, for a cylindrical air-filled cavity, the normal derivative of the total wave displacement vanishes at the surface S, (i.e., a stress-free boundary condition). Then the total wave displacement, u^t, at the surface S satisfies the integral equation

$$-\frac{\partial u^{(i)}(r)}{\partial \hat{r}} = \frac{\partial}{\partial \hat{r}} \int_S u^{(t)}(r') \frac{\partial G(r,r')}{\partial \hat{r}'} \, dS \ ; \text{ for } r \text{ on } S \ . \qquad \text{(B-8)}$$

After the value for $u^{(t)}(r)$ at the surface $r = a$ is found, the total wave displacement for values of $r > a$ can be determined using the integral equation for r in V given by

$$u^{(t)}(r) = u^{(i)}(r) + \int_S u^{(t)}(r') \frac{\partial G(r,r')}{\partial \hat{r}'} \, dS \ . \qquad \text{(B-9)}$$

APPENDIX C

FORMAL SOLUTION OF THE INTEGRAL P_{nm}

APPENDIX C

Formal Solution of the Integral P_{nm}

The integral P_{nm} given in Equation (107a) may be transformed into a more convenient form for numerical evaluation. This integral can be converted to an integral having limits of integration between 0 to ∞ using the transformation

$$\int_{-\infty}^{\infty}(\ldots)e^{j\theta n}d\lambda = \int_{0}^{\infty}(\ldots)j^{n}\left[\left(\frac{\lambda + \sqrt{\lambda^{2} - k_{2}^{2}}}{k_{2}}\right)^{n} + \left(\frac{-\lambda + \sqrt{\lambda^{2} - k_{2}^{2}}}{k_{2}}\right)^{n}\right]d\lambda \ . \qquad \text{(C-1)}$$

Thus, the integral P_{nm} can be reduced to

$$P_{nm} = \frac{1}{j\pi}\int_{0}^{\infty}\varepsilon_{2}(\eta)\,\frac{e^{-2k_{2}(\eta^{2}-1)^{1/2}(H-h)}}{(\eta^{2}-1)^{1/2}}\left[\left(\eta + \sqrt{\eta^{2}-1}\right)^{n+m} + \left(-\eta + \sqrt{\eta^{2}-1}\right)^{n+m}\right]d\eta \qquad \text{(C-2)}$$

where:

$$\varepsilon_{2}(\eta) = \frac{-R + e^{-2k_{2}(\eta^{2}-\nu^{2})^{1/2}}}{1 - Re^{-2k_{2}(\eta^{2}-\nu^{2})^{1/2}}}$$

in which

$$R = \frac{\xi^{2}(\eta^{2}-\nu^{2})^{1/2} - (\eta^{2}-1)^{1/2}}{\xi^{2}(\eta^{2}-\nu^{2})^{1/2} + (\eta^{2}-1)^{1/2}}$$

and

$$\xi^{2} = \frac{\mu_{1}}{\mu_{2}} \ ;$$

$$\nu = \frac{k_{1}}{k_{2}} \ ;$$

$$k_{2} = \frac{\omega}{c_{2}} \ .$$

The integral P_{nm} can be evaluated along the contour shown in Figure C-1. Branch points exist at (1,0) and (ν,0). The branch lines (0-1) and 0-ν) separate the regions where $\mathrm{Im}(\eta^2-\nu^2)^{1/2}$ and $\mathrm{Im}(\eta^2-1)^{1/2}$ are positive (above the η_R-axis) from the regions where both of the quantities are negative (below the η_R-axis). Note that $\mathrm{Re}(\eta^2-\nu^2)^{1/2}$ and $\mathrm{Re}(\eta-1)^{1/2}$ are always positive (the integrals will not converge otherwise). In addition, simple poles η^P are located at

$$1 - Re^{-2k_2(\eta^2-\nu^2)^{1/2}} = 0 . \qquad \text{(C-3)}$$

This resonance equation has real solutions in the interval $1 < \eta_R^{(P)} < \nu$ on the top Riemann sheet, and complex solutions, to be denoted by $\hat{\eta}^{(P)}$, for $\hat{\eta}_R^{(P)} < 1$ on the lower Riemann sheet. The former describes spectral modal fields trapped inside the top layer and which decay into the bottom layer, whereas the latter describes nonspectral leaky modal fields that transmit energy into the bottom layer (Kamel and Felsen, 1981).

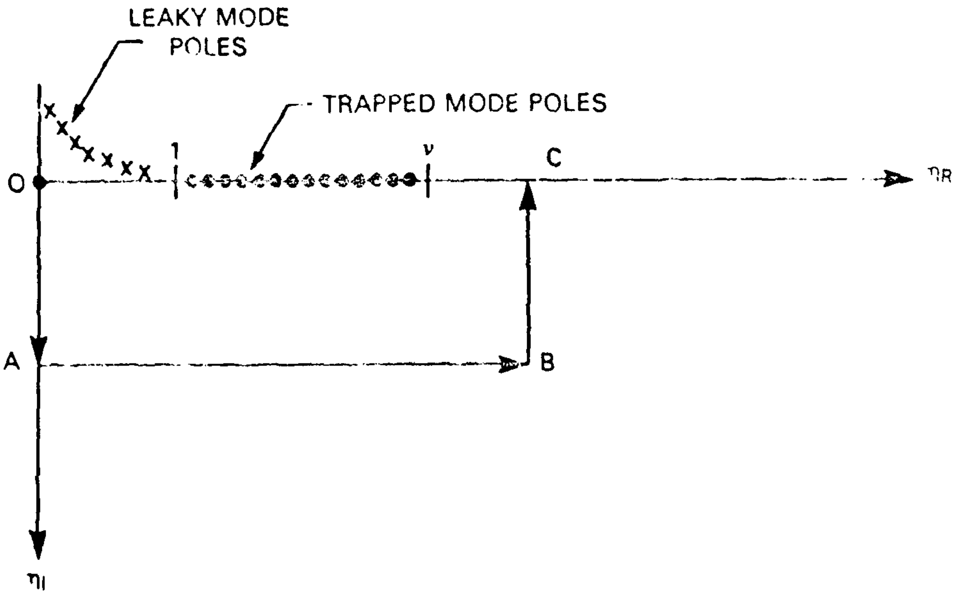


FIGURE C-1. CONTOUR OF INTEGRATION TO EVALUATE THE FUNCTION P_{nm}

Thus, the integral P_{nm} can be evaluated along the contour illustrated in Figure C-1 as follows

$$P_{nm} = \int_0^\infty (\quad) d\eta_R = \int_0^A (\quad) d\eta + \int_A^B (\quad) d\eta + \int_B^C (\quad) d\eta + \int_C^\infty (\quad) d\eta_R \qquad \text{(C-4)}$$

where:

$$\eta = \eta_R + j\eta_I .$$

This scheme of integration has been applied and tested using the integral form of the Hankel function by El-Akily (1980). In fact, to check the numerical evaluation of the integral, P_{nm}, the integral form of the Hankel function $H_n(2k_2|H-h|)$ may be used. This Hankel function can be deduced by making $\varepsilon_2(\eta) = 1$ in Equation (C-2); that is,

$$H_n(2k_2|H-h|) = \frac{1}{j\pi} \int_0^\infty \frac{e^{-2k_2(\eta^2-1)(H-h)}}{(\eta^2-1)^{1/2}} \left\{ \left(\eta + \sqrt{\eta 2-1}\right)^{n+m} + \left(-\eta + \sqrt{\eta 2-1}\right)^{n+m} \right\} d\eta \ . \qquad \text{(C-5)}$$

APPENDIX D

ASYMPTOTIC SOLUTION OF THE INTEGRAL P_{nm}

APPENDIX D

Asymptotic Solution of the Integral P_{nm}

Making the substitution $\lambda = k_2 \sin\theta$ in the integral P_{nm} given by Equation (107a) leads to

$$P_{nm} = (-j)^{n+m}\frac{1}{\pi}\int_{\Gamma}\varepsilon_2(\theta)e^{2jk_2(H-h)\cos\theta + j\theta(n+m)}d\theta \qquad (D-1)$$

where the integration path Γ is shown in Figure D-1 and $\varepsilon_2(\theta)$ is given by

$$\varepsilon_2(\theta) = \frac{-R + e^{2jk_2h(\nu^2-\sin^2\theta)^{1/2}}}{1 - Re^{2jk_2h(\nu^2-\sin^2\theta)^{1/2}}} \qquad (D-2)$$

in which

$$R = \frac{\xi^2(\nu^2-\sin^2\theta)^{1/2} - \cos\theta}{\xi^2(\nu^2-\sin^2\theta)^{1/2} + \cos\theta},$$

and

$$\xi^2 = \frac{\mu_1}{\mu_2};$$

and $\nu = \dfrac{k_1}{k_2}$.

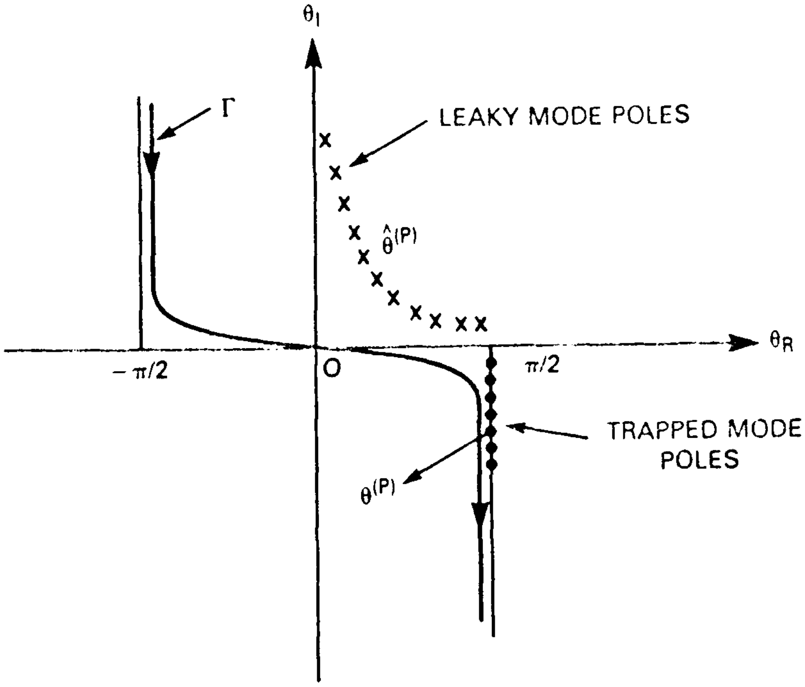


FIGURE D-1. INTEGRATION PATH FOR THE INTEGRAL P_{nm} IN THE COMPLEX-ANGLE (θ) PLANE

The asymtotic solution of the integral P_{nm} may be derived using the saddle-point method as described in Kong (1986). To obtain an appropriate form of the integral given in Equation (D-1) and to remove the singularities from the resonance equation, the integrand, $\varepsilon_2(\theta)$, may be expanded in a power series as

$$\varepsilon_2(\theta) = \frac{-R + e^{2jk_2h\alpha}}{1 - Re^{2jk_2h\alpha}} = \sum_{\ell=0}^{\infty}\left[R^{\ell}e^{2jk_2h\alpha(\ell+1)} - R^{\ell+1}e^{2jk_2h\alpha\ell}\right] \qquad \text{(D-3)}$$

where:

$$\alpha = (\nu^2 - \sin^2\theta)^{1/2} .$$

Equation (D-1) may then be expressed as

$$P_{nm} = (-1)^{n+m}\frac{1}{\pi}\sum_{\ell=0}^{\infty}\Big\{\int_{\Gamma} R^{\ell}e^{2jk_2h\alpha(\ell+1) + j\theta(n+m)}e^{2jk_2(H-h)\cos\theta}d\theta$$

$$- \int_{\Gamma} R^{\ell+1}e^{2jk_2\alpha h\ell + j\theta(n+m)}e^{2jk_2(H-h)\cos\theta}d\theta\Big\} . \qquad \text{(D-4)}$$

Each integral of Equation (D-4) may be represented in the form

$$I_\ell(\eta) = \int_\Gamma d\theta F_\ell(\theta)e^{\eta f(\theta)} \tag{D-5}$$

where η is a large parameter. Equation (D-5) has been derived by the saddle-point method and is (Kong, 1986)

$$I_\ell(\eta) = F_\ell(\theta_0)e^{\eta f(\theta_0)}\sqrt{\frac{2\pi}{-\eta f''}}\left\{1 + \frac{1}{2\eta f''}\left[\frac{f'''}{f''}\frac{F'}{F} + \frac{1}{4}\frac{f^{iv}}{f''} - \frac{5}{12}\frac{(f''')^2}{(f'')^2} - \frac{F''}{F}\right] + \dots\right\} \tag{D-6}$$

where θ_0 is the saddle point.

From Equation (D-4), the function, f, for the first integral is

$$f_{\ell+1} = j\left[\cos\theta + \frac{\alpha h(\ell+1)}{H-h}\right]$$

and is, for the second integral

$$f_\ell = j\left[\cos\theta + \frac{\alpha h\ell}{H-h}\right]$$

from which the parameter η is deduced to be

$$\eta = 2k_2(H-h) .$$

Since the integrals of Equation (D-4) have the same form, the saddle point, θ_0, is determined for the first integral to be

$$\frac{\partial}{\partial\theta} j\left[\cos\theta + \frac{\alpha h(\ell+1)}{H-h}\right] = 0$$

which gives

$$\theta = n\pi;\ \text{for } n = 0, \pm 1, \pm 2, \dots\dots$$

Since the saddle point must be within the interval $-\pi/2 < \theta_o < \pi/2$, then θ_o must be equal to zero. Therefore, evaluating the derivatives of the function $f \equiv f_{\ell+1}(\theta)$ at $\theta_o = 0$ and substituting them into Equation (D-6) yields

$$I_\ell(\eta) = F_\ell(\theta_o)e^{2jk_2(H-h)[1 + \frac{\nu h(\ell+1)}{H-h}]} \sqrt{\frac{2\pi}{2jk_2[1 + \frac{h(\ell+1)}{(H-h)\nu}]}}$$

$$\times \left[1 - \frac{j}{4k_2(H-h)\left[1 + \frac{h(\ell+1)}{(H-h)\nu}\right]}\right]\left[\frac{1}{4} \frac{1 + \frac{4}{\nu} e_{\ell+1}(1 - \frac{3}{4\nu^2}}{1 + \frac{e_{\ell+1}}{\nu}} + \frac{F''}{F}\right] \qquad \text{(D-7)}$$

where:

$$e_{\ell+1} = \frac{h(\ell+1)}{H-h}$$

and $F_\ell(\theta_o)$, $(F''/F)_{\theta=\theta_o}$ are yet to be determined.

For convenience, the function $F_\ell = F_\ell(\theta)$ is defined as

$$F_\ell = R^a e^{\theta c} \qquad \text{(D-8)}$$

where:

$$R \equiv R(\theta) .$$

In order to evaluate Equation (D-7) for $\theta_o = 0$, the function F_ℓ and the ratio F_ℓ''/F_ℓ are required. The second derivative of Equation (D-8) is derived to be

$$F_\ell'' = e^{\theta c} \left\{\frac{d^2R_a}{d\theta^2} + 2c \frac{dR^a}{d\theta} + R^a c^2\right\} \qquad \text{(D-9)}$$

where:

$$\frac{dRa}{d\theta} = aRa-1 \frac{dR}{d\theta} ;$$

$$\frac{d^2R^a}{d\theta^2} = a(a-1)R^{a-1}\left(\frac{dR}{d\theta}\right)^2 + aR^{a-1} \frac{d^2R}{d\theta^2} ;$$

$$\frac{dR}{d\theta} = \frac{2(\zeta^2\alpha)\ \sin\ \theta}{(\zeta^2\alpha\ +\ \cos\ \theta)^2}\ ;$$

$$\frac{d^2R}{d\theta^2} = 2(\zeta^2\alpha)\ \frac{[\zeta^2\alpha\ \cos\ \theta\ +\ \sin^2\theta\ +\ 1]}{(\zeta^2\alpha\ +\ \cos\ \theta)}\ .$$

At the saddle-point $\theta_o = 0$, the ratio between Equation (D-9) and D-8) is

$$\left(\frac{F_\ell''}{F_\ell}\right)_a = \frac{2\zeta^2 va}{(\zeta^2 v)^2 - 1} + c^2 \qquad \text{(D-10)}$$

and the function $F_\ell(0)$ is given by

$$(F_\ell)_a = R^a\ .$$

In the first integral of Equation (D-4), defined as

$$I_1 = \sum_{\ell=0}^{\infty} \int_\Gamma R^\ell e^{2k_2(H-h)[\cos\ \theta\ +\ \frac{\alpha h(\ell+1)}{H-h}]\ +\ j\theta(n+m)} d\theta\ , \qquad \text{(D-11)}$$

the function F_ℓ at saddle point $\theta_o = 0$ is

$$F_\ell(0) = R^\ell \qquad \text{(D-12)}$$

and

$$\left(\frac{F_\ell''}{F_\ell}\right)_{\theta_o=0} = \frac{2\zeta^2 v\ell}{(\zeta^2 v)^2 - 1} - (n+m)^2\ . \qquad \text{(D-13)}$$

Thus, the asymtotic solution of the integral I_1 is

$$I_1 \sim \sum_{\ell=0}^{\infty} e^{2jk_2(H-h)\ +\ 2jk_2h(\ell+1)v\ -\ \frac{\pi}{4}\,j} \left[\frac{2\pi}{2k_2\{(h-h)\ +\ h\ \frac{(\ell+1)}{v}\}}\right]^{1/2}$$

$$\times\ R^\ell \left[1 - \frac{j}{4k_2[\ (H-h)\ +\ \frac{h(\ell+1)}{v}]}\right]\left[\frac{1}{4} + \frac{2\zeta^2 v\ell}{(\zeta^2 v)^2 - 1} - (n+m)^2\right] \qquad \text{(D-14)}$$

and

$$I_2 \sim \sum_{\ell=0}^{\infty} e^{2jk_2(H-h) + 2jk_2 h\ell\nu - \frac{\pi}{4} j} \left[\frac{2\pi}{2k_2\left[(H-h) + \frac{h\ell}{\nu}\right]}\right]^{\frac{1}{2}}$$

$$\times R^{\ell+1}\ 1 - \frac{j}{4k_2\left[(H-h) + \frac{h\ell}{\nu}\right]} \left[\frac{1}{4} + \frac{2\zeta^2\nu(\ell+1)}{(\zeta^2\nu)^2 - 1} + (n+m)^2\right] \quad . \qquad \text{(D-15)}$$

Finally, replacing the asymptotic solutions of the integrals I_1 and I_2 in Equation (D-4) leads to the asymptotic solution of the integral, P_{nm}. That is,

$$P_{nm} \cong \sum_{\ell=0}^{\infty} e^{2jk_2[H-h+h(\ell+1)\nu] - \frac{\pi}{4} j\ \frac{\pi}{2} (n+m)j}$$

$$\times \left[\frac{2}{2k_2\left[(H-h) + \frac{(\ell+1)h}{\nu}\right]}\right]^{\frac{1}{2}} R^{\ell} \left[1 - \frac{j}{4k_2\left[(H-h) + \frac{h(\ell+1)}{\nu}\right]}\right] \left[\frac{1}{4} + \frac{2\zeta^2\nu\ell}{(\zeta^2\nu)^2 - 1} - (n+m)^2\right]$$

$$- \sum_{\ell=o}^{\infty} e^{2jk_2[H-h+h\ell\nu] - \frac{\pi}{4} j\ \frac{\pi}{2} (n+m)j} \left[\frac{2}{2k_2\left[H-h + \frac{\ell h}{\nu}\right]}\right]^{\frac{1}{2}}$$

$$\times R^{\ell+1} \left[1 - \frac{j}{4k_2\left[(H-h) + \frac{h\ell}{\nu}\right]}\right] \left[\frac{1}{4} + \frac{2\zeta^2\nu(\ell+1)}{(\zeta^2\nu)^2 - 1} - (n+m)^2\right] \quad ,$$

$$\text{for } c_1\rho < \rho_2 c_2 \ . \qquad \text{(D-16)}$$

In the optical limit as

$$2k_2[(H-h)] \rightarrow \infty \qquad .$$

the following approximate expression for the integral P_{nm} may be obtained in terms of Hankel functions. That is,

$$P_{nm} \sim \sum_{\ell=0}^{\infty} a_{\ell+1} H_{n+m}\{2k_2[H-h+h(\ell+1)\nu]\} R^{\ell}$$

$$- \sum_{\ell=0}^{\infty} b_{\ell} H_{n+m}\{2k_2[(H-h) + h\ell\nu]\} R^{\ell+1} \qquad \text{(D-17)}$$

where:

$$a_{\ell+1} = \left[\frac{1 + \frac{h(\ell+1)\nu}{H-h}}{1 + \frac{h(\ell+1)}{(H-h)\nu}} \right]^{\frac{1}{2}} ;$$

and

$$b_{\ell} = \left[\frac{1 + \frac{h\ell\nu}{H-h}}{1 + \frac{h\ell}{(H-h)\nu}} \right]^{\frac{1}{2}} .$$

In order to simplify numerical calculations, the approximate solution of P_{nm} given by Equation (D-17) may be written as

$$P_{nm} \sim - \frac{\zeta^2\nu - 1}{\zeta^2\nu + 1} H_{n+m}[2k_2(H-h)]$$

$$+ \sum_{\ell=1}^{\infty} a_{\ell} \left[\frac{\zeta^2\nu - 1}{\zeta^2\nu + 1}^{\ell-1} - \frac{\zeta^2\nu - 1}{\zeta^2\nu + 1}^{\ell+1} \right] H_{n+m}(k_2 s_{\ell}) \qquad \text{(D-18)}$$

in which

$$\zeta^2 = \mu_1/\mu_2 \ ;$$

$$\nu = k_1/k_2 \ ;$$

$$s_\ell = 2(H-h+h\ell\nu) \ ;$$

and

$$a_\ell = \left[\frac{1 + h\ell\nu/(H-h)}{1 + h\ell/(H-h)\nu}\right]^{\frac{1}{2}} \ .$$

APPENDIX E

SH-WAVE SEISMIC PULSES RADIATED BY A LINE SOURCE IN A TWO-LAYER HALF-SPACE

APPENDIX E

SH-Wave Seismic Pulses Radiated by a Line Source in a Two-Layer Half-Space

E.1 The Incident Field and Its Representation

The transient SH-wave displacement for a line source located at the source position (x_s, y_s) in the bedrock semi-infinite half-space satisfies the wave equations:

$$\mu_1 \nabla^2 \hat{u} = \rho_1 \frac{\partial^2 \hat{u}}{\partial t^2} \; ; \text{ for } 0 < x < h \tag{E-1}$$

and

$$\mu_2 \nabla^2 \hat{u} - \rho_2 \frac{\partial^2 \hat{u}}{\partial t^2} = -\delta(x-x_s)\delta(y-y_s)\delta(t) \; ; \text{ for } x > h \; . \tag{E-2}$$

The transient solution for cylindrical wave motion may be developed using the Cagniard-deHoop approach described in deHoop (1960) and in Aki and Richards (1980). This solution employs the Laplace transform of the SH-wave displacement for a line source in an unbounded medium

$$u(x,y,x_s,y_s,S) = \frac{1}{2\pi\rho c} \, \mathrm{Im}\left\{\int_0^{j\infty} \frac{e^{-S[p(y-y_s) + \alpha(x-x_s)]}}{\alpha} dp\right\} \; , \tag{E-3}$$

where:

$$\alpha = \left(\frac{1}{c^2} - p^2\right)^{1/2} \; ;$$

$$R = [(x-x_s)^2 + (y-y_s)^2]^{1/2} \; ; \text{ and}$$

c = the shear wave velocity.

The branch cuts $\pm\frac{1}{c}$ determined by Re $\alpha > 0$ and the path $p = p(t)$ for different values of t are shown in Figure E-1. The function $p(t)$ is obtained by solving the equation

$$t = p(y-y_s) + \alpha(x-x_s) \; .$$

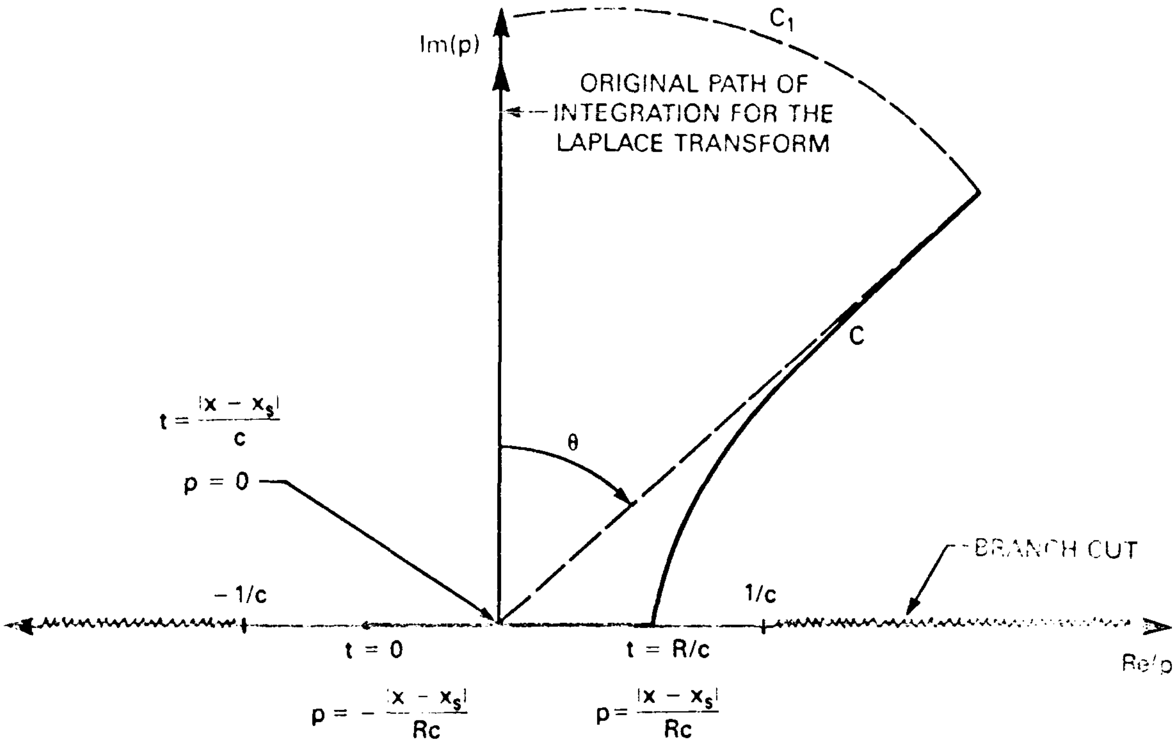


FIGURE E-1. BRANCH CUTS, THE INTEGRATION PATH FOR THE LAPLACE TRANSFORM, AND THE CAGNIARD PATH C IN THE COMPLEX p-PLANE

The Cagniard path C in the complex p-plane is obtained for which the quantity $p(y-y_s) + \alpha(x-x_s)$ is real. The path begins at t = 0 on the negative real p-axis, turns on the branch of a hyperbola at t = R/c and continues on a path having an asymptote at an angle $\theta = \tan^{-1}(y-y_s)/|x-x_s|$ with respect to the positive imaginary p-axis. Note that on the real p-axis the path C cannot lie on the branch cuts. Figure E-1 also shows part of a large arc, C_1, in the first quadrant connecting the imaginary p-axis and the path C. The angle θ may be identified as the angle of incidence of the ray from the source to a detector.

There is no contribution from the integrand of Equation (E-3) along that part of the path C between t = 0 and t = R/c, since p is purely real. Neither is there a constribution from the large arc C_1 in the first quadrant. Since there are no singularities between the path C and the positive imaginary p-axis, Aki and Richards (1980) conclude that the Laplace transform of the SH-wave displacement for a line source in an unbounded medium, in terms of the Cagniard path C, is

$$u(x,y,x_s,y_s,S) = \frac{1}{2\pi\rho c^2} \operatorname{Im}\{\int_C \frac{e^{-S[p(y-y_s) + \alpha(x-x_s)]}}{\alpha} dp \ . \qquad \text{(E-4)}$$

E.2 Transient SH-Wave Displacements in a Two-Layer Half-Space

The integral representation of the wave displacement given by Equation (E-4) may be generalized for the SH-wave displacement in regions of different elastic properties. Thus,

For the region $0 \leqslant x \leqslant h$:

$$u_1 = \frac{1}{2\pi\rho_2 c_2^2}\, \mathrm{Im}\left\{\int_C \left[f_1(p)e^{-S\alpha_1 x} + g_1(p)e^{S\alpha_1 x}\right]\frac{e^{-Sp(y-y_s)}}{\alpha_1}\,dp\right\} \qquad \text{(E-5)}$$

For the region $x > h$:

$$u_2 = \frac{1}{2\pi\rho_2 c_2^2}\, \mathrm{Im}\left\{\int_C \left[e^{-S\alpha_2|x-x_s|} + f_2(p)e^{-S\alpha_2 x}\right]\frac{e^{-Sp(y-y_s)}}{\alpha_2}\,dp\right\} \qquad \text{(E-6)}$$

where:

$$\alpha_1 = \left(\frac{1}{c_1^2} - p^2\right)^{1/2} \;;$$

and

$$\alpha_2 = \left(\frac{1}{c_2^2} - p^2\right)^{1/2} \;.$$

The factors $f_1(p)$, $g_1(p)$, and $f_2(p)$ are arbitrary functions of p which render the integrals convergent and satisfy the following boundary conditions:

At the surface:

$$\left(\mu_1 \frac{\partial u_1}{\partial x}\right)_{x=0} = 0 \;.$$

At the layer interface:

$$\left(\mu_1 \frac{\partial u_1}{\partial x}\right)_{x=h} = \left(\mu_2 \frac{\partial u_2}{\partial x}\right)_{x=h}$$

and

$$(u_1)_{x=h} = (u_2)_{x=h} \;. \qquad \text{(E-7)}$$

These boundary conditions lead to three linear algebraic equations to solve for the unknown coefficients $f_1(p)$, $g_1(p)$, and $f_2(p)$. The solution of the SH-wave displacements for the surface layer and the cavity host medium is:

For the region $0 < x < h$:

$$u_1(S) = \frac{1}{2\pi\rho_2 c_2^2} \operatorname{Im}\{\int_C \varepsilon_1(p) \frac{e^{-S[p(y-y_s) + \alpha_2(x_s-h))]}}{\alpha_2}$$

$$\times \left[e^{-S\alpha_1(x+h)} + e^{-S\alpha_1(h-x)}\right]dp\} \qquad \text{(E-8)}$$

For the region $x > h$:

$$u_2(S) = \frac{1}{2\pi\rho_2 c_2^2} \operatorname{Im}\{\int_C \frac{e^{-S[p(y-y_s) + \alpha_2(x-x_s)]}}{\alpha_2} dp\}$$

$$+ \frac{1}{2\pi\rho c_2^2} \operatorname{Im}\{\int_C \frac{\varepsilon_2(p)}{\alpha_2} e^{-S[p(y-y_s) + \alpha_2(x-x_s)]} dp\} \qquad \text{(E-9)}$$

where:

$$\varepsilon_1(p) = \frac{T(p)}{1-R(p)e^{-2S\alpha_1 h}} ;$$

$$\varepsilon_2(p) = \frac{-R(p) + e^{-2S\alpha_1 h}}{1 - R(p)e^{-2S\alpha_1 h}} ; \qquad \text{(E-10)}$$

$$T(p) = \frac{2\mu_2\alpha_2}{\mu_1\alpha_1 + \mu_2\alpha_2} ;$$

$$R(p) = \frac{\mu_1\alpha_1 - \mu_2\alpha_2}{\mu_1\alpha_1 + \mu_2\alpha_2} . \qquad \text{(E-11)}$$

E.2.1 Detector in the Bedrock

The first integral of Equation (E-9) represents the incident SH-wave displacement, $u_2^{(i)}(S)$, for a line source in an unbounded medium having an SH-wave velocity, c_2. Applying an inversion technique developed by deHoop (1960) and presented by Aki and Richards (1980), the integrand of the incident SH-wave displacement, $u_2^{(i)}(S)$, may be inverted in terms of the time variable of integration, t, to yield

$$\hat{u}_2^{(i)}(S) = \frac{1}{2\pi\rho_2 c_2^2} \int_{R_s/c_2}^{\infty} \frac{e^{-St}}{(t^2 - R_s^2/c_2^2)^{1/2}}\, dt \qquad \text{(E-12)}$$

from which the delta-function response can be identified and is given by

$$\hat{u}_2^{(i)}(t) = \frac{1}{2\pi\rho_2 c_2^2} \frac{H(t-t_s)}{(t^2-t_s^2)^{1/2}} \quad ; \text{ for } t > t_s \qquad \text{(E-13)}$$

where:

$$R_s = [(x-x_s)^2 + (y-y_s)^2]^{1/2} \quad ; \text{ and}$$

$t_s = R_s/c_s$ = arrival time at the detector location.

The second term of Equation (E-9), $u_2^{refl}(S)$, gives the generalized reflection response in the bedrock medium. The function $\varepsilon_2(p)$ may be expressed as a power series expansion of the denominator

$$\frac{1}{1 - Re^{-2Sh\alpha_1}} = \sum_{\ell=0}^{N} R^{\ell} e^{-2Sh\ell\alpha_1} \qquad \text{(E-14)}$$

to yield,

$$\varepsilon_2(p) = \sum_{\ell=0}^{N} R^{\ell} e^{-2S\alpha_1 h(\ell+1)} - \sum_{\ell=0}^{N} R^{\ell+1} e^{-2S\alpha_1 h\ell} \ , \qquad \text{(E-15)}$$

where N is the total number of reflections within the surface layer. Substituting Equation (E-15) into the second term of Equation (E-9), the generalized reflection is

$$u_2^{refl}(S) = \frac{1}{2\pi\rho_2 c_2^2} \sum_{\ell=0}^{N} Im\int_{t_{\ell+1}}^{\infty} \left[\frac{R^{\ell}(p)}{\alpha_2} \frac{dp}{dt}\right]_{p=p_{\ell+1}(t)} e^{-St} \, dt$$

$$- \frac{1}{2\pi\rho_2 c_2^2} \sum_{\ell=0}^{N} Im\int_{t_{\ell}}^{\infty} \left[\frac{R^{\ell+1}(p)}{\alpha_2} \frac{dp}{dt}\right]_{p=p_{\ell}(t)} e^{-St} \, dt \, . \qquad (E-16)$$

In the time domain, the generalized reflection displacement in the delta-function response can be identified directly from Equation (E-16) and is thereby

$$\hat{u}_2^{refl}(t) = \frac{1}{2\pi\rho_2 c_2^2} \left\{ \sum_{\ell=0}^{N} Im\left[\frac{R^{\ell}(p)}{\alpha_2} \frac{dp}{dt}\right]_{p=p_{\ell+1}(t)} H(t-t_{\ell+1}) \right.$$

$$\left. - \sum_{\ell=0}^{N} Im\left[\frac{R^{\ell+1}(p)}{\alpha_2} \frac{dp}{dt}\right]_{p=p_{\ell}(t)} H(t-t_{\ell}) \right\} \, , \qquad (E-17)$$

where $p_\ell(t)$ and $p_{\ell+1}(t)$ are solutions of the nonlinear equations given by

$$t = p(y-y_s) + \alpha_2(x+x_s-2h) + 2\alpha_1 h\ell \qquad (E-18)$$

and

$$t = p(y-y_s) + \alpha_2(x+x_s-2h) + 2\alpha_1 h(\ell+1) \, , \qquad (E-19)$$

respectively. The time derivatives of the variable, p, associated with Equations (E-18) and (E-19) are given by

$$\frac{dp_\ell}{dt} = \frac{1}{|y-y_s| - \frac{(x+x_s-2h)p_\ell}{\alpha_2(\ell)} \quad \frac{2h\ell p_\ell}{\alpha_1(\ell)}} \qquad (E-20)$$

and

$$\frac{dp_{\ell+1}}{dt} = \frac{1}{|y-y_s| - \frac{(x+x_s-2h)p_{\ell+1}}{\alpha_2(\ell+1)} - \frac{2h(\ell+1)p_{\ell+1}}{\alpha_1(\ell+1)}} \qquad (E-21)$$

in which

$$\alpha_j(\ell) = \left(\frac{1}{c_j^2} - p_\ell^2\right)^{1/2}$$

$$\alpha_j(\ell+1) = \left(\frac{1}{c_j^2} - p_{\ell+1}^2\right)^{1/2} \quad ; \text{ for } j = 1,2$$

The times of arrival associated with the first and the second terms of Equation (E-17) are given by

$$t_{\ell+1} = \frac{x+x_s-2h}{c_2 \cos \theta_2^{(\ell+1)}} + \frac{2h(\ell+1)}{c_1 \cos \theta_1^{(\ell+1)}} \tag{E-22}$$

and

$$t_\ell = \frac{x+x_s-2h}{c_2 \cos \theta_2^{(\ell)}} + \frac{2h\ell}{c_1 \cos \theta_1^{(\ell)}} \tag{E-23}$$

where the angles θ_1 and θ_2 are the angles of the ray trajectories in the surface layer and in the semi-infinite half-space, respectively. Each term of Equation (E-17) represents a ray path that reaches the observation point after ℓ reflections in the surface layer. In fact, the multiples occurring within the surface layer are retained and, after ℓ reflections, the SH wave is transmitted into the bedrock formation. The ray contribution associated with the first term of Equation (E-17) is shown in Figure E-2(a). The horizontal distance between source and detector can be inferred directly from the ray geometry shown in Figure E-2(a) as

$$r_{\ell+1} = (x+x_s-2h) \tan \theta_2^{(\ell+1)} + 2h(\ell+1) \tan \theta_1^{(\ell+1)} \, . \tag{E-24}$$

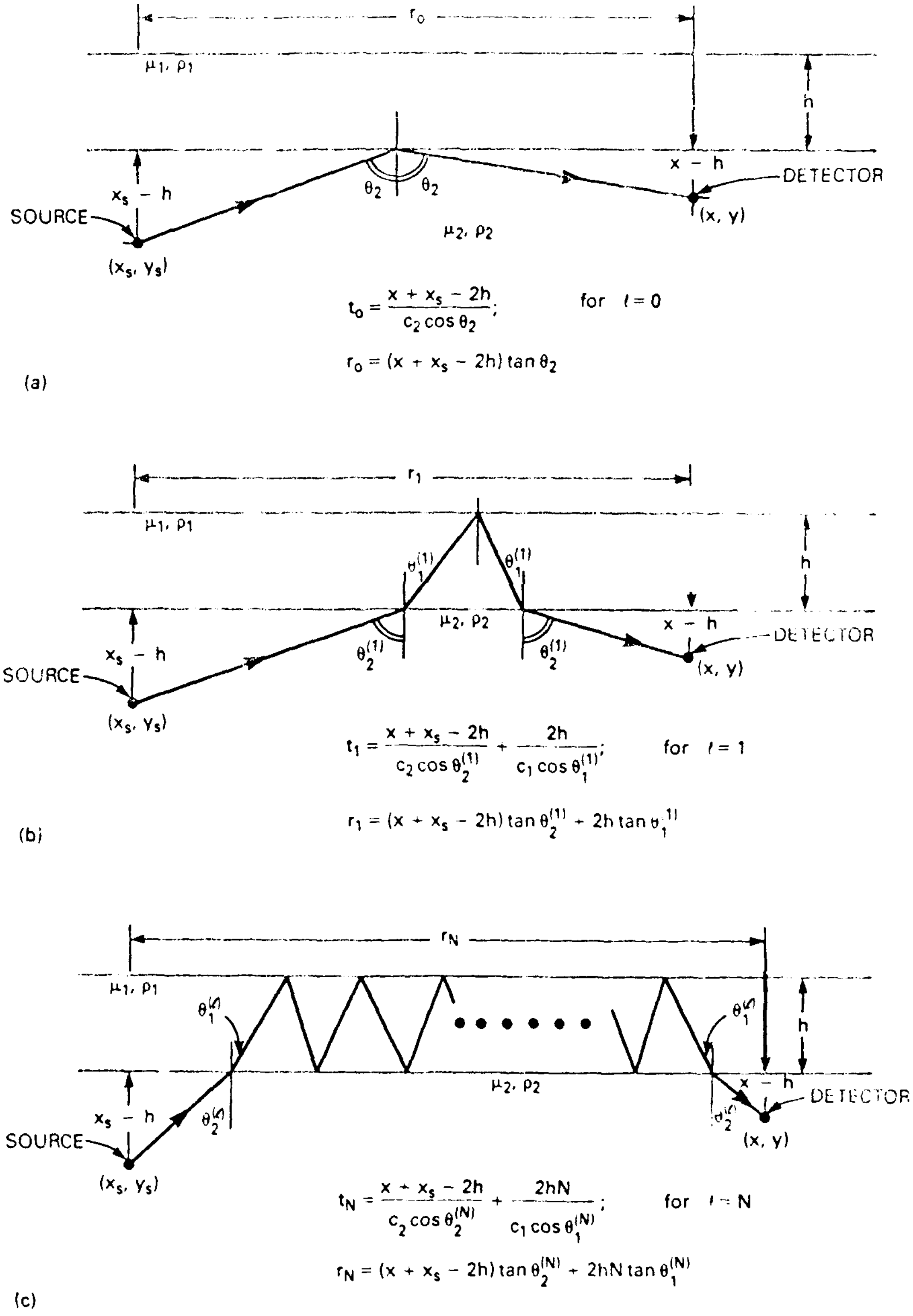
Similarly, the horizontal distance between the source and detector for the second term of Equation (E-17) is given by

$$r_\ell = (x+x_s-2h) \tan \theta_2^{(\ell)} + 2h\ell \tan \theta_1^{(\ell)} \, . \tag{E-25}$$

The angles $\theta_1^{(\ell)}$ and $\theta_2^{(\ell)}$ may be obtained for an arbitrary point of observation p(x,y) by solving the following systems of equations:

$$r_\ell = (x+x_s-2h) \tan \theta_2^{(\ell)} + 2h\ell \tan \theta_1^{(\ell)}$$

$$\frac{\sin \theta_2^{(\ell)}}{c_2} = \frac{\sin \theta_1^{(\ell)}}{c_1} = p^{(\ell)} \tag{E-26}$$



$$t_0 = \frac{x + x_s - 2h}{c_2 \cos\theta_2}; \qquad \text{for } \ell = 0$$

$$r_0 = (x + x_s - 2h)\tan\theta_2$$

$$t_1 = \frac{x + x_s - 2h}{c_2 \cos\theta_2^{(1)}} + \frac{2h}{c_1 \cos\theta_1^{(1)}}; \qquad \text{for } \ell = 1$$

$$r_1 = (x + x_s - 2h)\tan\theta_2^{(1)} + 2h\tan\theta_1^{1)}$$

$$t_N = \frac{x + x_s - 2h}{c_2 \cos\theta_2^{(N)}} + \frac{2hN}{c_1 \cos\theta_1^{(N)}}; \qquad \text{for } \ell = N$$

$$r_N = (x + x_s - 2h)\tan\theta_2^{(N)} + 2hN\tan\theta_1^{(N)}$$

FIGURE E-2. GEOMETRY OF SH-WAVE RAY PATHS FROM A LINE SOURCE TO A DETECTOR IN THE BEDROCK LAYER SHOWING THE CONTRIBUTIONS ASSOCIATED WITH THE POWER SERIES EXPANSION OF EQUATION (E-17)

and

$$r_{\ell+1} = (x+x_s-2h) \tan \theta_2^{(\ell+1)} + 2h(\ell+1) \tan \theta_1^{(\ell+1)}$$

$$\frac{\sin \theta_2^{(\ell+1)}}{c_2} = \frac{\sin \theta_1^{(\ell+1)}}{c_1} = p^{(\ell+1)} \text{ .} \qquad \text{(E-27)}$$

These equations may be expressed in terms of the ray-path parameter, p, as

$$r_\ell = \left. \frac{(x+x_s-2h)p}{\sqrt{c_2^{-2} - p^2}} + \frac{2h\ell p}{\sqrt{c_1^{-2} - p^2}} \right|_{p=p^{(\ell)}} \qquad \text{(E-28)}$$

and

$$r_{\ell+1} = \left. \frac{(x+x_s-2h)p}{\sqrt{c_2^{-2} - p^2}} + \frac{2h(\ell+1)p}{\sqrt{c_1^{-2} - p^2}} \right|_{p=p^{(\ell+1)}} \qquad \text{(E-29)}$$

$$\text{for } 0 < p < \frac{1}{c_2} \text{ .}$$

For a prescribed source-detector position, Equations (E-28) and (E-29) must be solved numerically for the ray parameter p. In this way the pairs of angles of incidences $[\theta_1^{(\ell)}, \theta_2^{\ell}]$ and $[\theta_1^{(\ell+1)}, \theta_2^{(\ell+1)}]$ may be determined and substituted in Equations (E-22) and (E-23) to obtain the arrival times, t_ℓ and $t_{\ell+1}$, for ℓ = 0,1,2,... N.

E.2.1.1 Numerical Algorithm to Solve the Equation $p = p(t)$

Since no simple analytical relationships exist either to solve Equation (E-18) or (E-19), the solution of $p = p(t)$ must be determined with the aid of a numerical technique. An iterative technique can be applied successfully if the starting value is chosen judiciously (deHoop, 1979). Starting values of $p = P$ may be obtained from the equation

$$P|y-y_s| + (1/C^2-P^2)^{1/2}H = t \qquad \text{(E-30)}$$

where the unknown quantities H and C are selected such that Equation (E-30) approximates that of Equation (E-18) or (E-19). In particular, values C and H are chosen such that Equation (E-30) asymptotically coincides with Equation

(E-18) or (E-19), as $|p| \rightarrow \infty$ and at $p = 0$. For example, Equation (E-18) may be expressed in terms of the parameter p as

$$p|y-y_s| + (x+x_s-2h)\left(\frac{1}{c_2{}^2} - p^2\right)^{1/2} + 2h\ell\left(\frac{1}{c_1{}^2} - p^2\right) = t$$

which, for large values of $|p|$, reduces to

$$p \cong t/[(y-y_s)^2 + j(x+x_s-2h+2h\ell)] \; . \qquad \text{(E-31)}$$

In the same manner, for large values of $|P|$, Equation (E-30) reduces to

$$P \cong t/[(y-y_s) + jH] \; . \qquad \text{(E-32)}$$

Comparing Equations (E-31) and (E-32), it follows that

$$H = x+x_s + 2h(\ell-1) \; .$$

The second conditions for Equations (E-30) and (E-31) are

$$p = 0 \text{ at } t = \frac{x+x_s-2h}{c_2} + \frac{2h\ell}{c_1}$$

and

$$P = 0 \text{ at } t = H/C \; ,$$

from which

$$\frac{1}{C} = \frac{(x+x_s-2h)/c_2 + 2h\ell/c_1}{x+x_s+2h(\ell-1)} \; .$$

Solving Equation (E-28), the starting value P_ℓ to be used for p_ℓ in Equation (E-18) to begin the iteration process is, therefore

$$P_\ell = \frac{|y-y_s|}{|y-y_s|^2 + H_\ell{}^2}\, t + \frac{jH_\ell}{|y-y_s|^2 + H_\ell{}^2}\left[t^2 - \frac{(y-y_s)^2 + H_\ell{}^2}{C_\ell{}^2}\right]^{1/2} \qquad \text{(E-33)}$$

$$\text{for} \quad [(y-y_s)^2 + H_\ell{}^2]^{\frac{1}{2}}/C_\ell < t < \infty \; ,$$

where:

$$H_\ell = x+x_s+2h(\ell-1) \quad ; \text{ and}$$

$$1/C_\ell = (\frac{x+x_s-2h}{c_2} + \frac{2h\ell}{c_1})/[x+x_s+2h(\ell-1)] \quad .$$

Alternatively, the starting value $P_{\ell+1}$ to be used for $p_{\ell+1}$ to begin Equation (E-19) in the iterative numerical procedure is

$$P_{\ell+1} = \frac{|y-y_s|}{|y-y_s|^2 + H^2_{\ell+1}} t + \frac{jH_{\ell+1}}{|y-y_s|^2 + H^2_{\ell+1}} [t^2 - \frac{(y-y_s)^2 + H^2_{\ell+1}}{C^2_{\ell+1}}]^{1/2} \qquad \text{(E-34)}$$

$$\text{for} \quad [|y-y_s|^2 + H_{\ell+1}{}^2]^{\frac{1}{2}}/C_{\ell+1} < t < \infty \; ;$$

where:

$$H_{\ell+1} = x+x_s+2h\ell \quad ; \text{ and}$$

$$1/C_{\ell+1} = [\frac{x+x_s-2h}{c_2} + \frac{2h(\ell+1)}{c_1}]/(x+x_s+2h\ell) \quad .$$

For an initial value of $\underline{t} > t_\ell$, the starting parameter, P, is determined from Equation (E-31) and then substituted into Equation (E-18) such that the following requirements for the Cagniard contours are satisfied. That is,

$$\text{Re}[\,p(y-y_s) + (x+x_s-2h)(\frac{1}{c_2{}^2} - p^2)^{1/2} + 2h\ell(\frac{1}{c_1{}^2} - p^2)^{1/2}] = t$$

$$\text{Im}[\,p(y-y_s) + (x+x_s-2h)(\frac{1}{c_2{}^2} - p^2)^{1/2} + 2h\ell(\frac{1}{c_1{}^2} - p^2)^{1/2}] = 0$$

where p is complex and t is real.

For example, for an initial parameter $p = p_1 + jp_2$ and a given value of t greater than the arrival time, t_ℓ, the equations

$$f_A = p_1|y-y_s| + \text{Re}[g_\ell(p)] - t \qquad \text{(E-36)}$$

$$f_B = p_2|y-y_s| + \text{Im}[g_\ell(p)]$$

in which

$$g_\ell(p) = \left(x+x_s-2h\right)\left(\frac{1}{c_2{}^2} - p^2\right)^{1/2} + 2h\ell\left(\frac{1}{c_1{}^2} - p^2\right)^{1/2}$$

may be implemented in a numerical procedure for solving a system of nonlinear equations for the unknowns p_1 and p_2. Thus, for given values of t and starting values of the complex variable, p, a Cagniard contour may be constructed and used to evaluate Equation (E-17).

E.2.1.2 First Motion Approximation

In order to simplify the analysis, Equation (E-17) may be written with only one summation as

$$\hat{u}_2{}^{ref}(t) = \frac{1}{2\pi\rho_2 c_2{}^2} \, \mathrm{Im}\left[\frac{R(p)}{\alpha_2}\frac{dp}{dt}\right]_{p=p_o(t)} H(t-t_o)$$

$$+ \sum_{\ell=1}^{N} \mathrm{Im}\left[\left(R^{-1} - R\right)\frac{R^\ell}{\alpha_2}\frac{dp}{dt}\right]_{p=p_\ell(t)} H(t-t_\ell) \, , \qquad \text{(E-37)}$$

where $p_o(t)$ and $p_\ell(t)$ are solutions of the equations

$$t = p(y-y_s) + \alpha_2(x+x_s-2h) \qquad \text{(E-38a)}$$

and

$$t = p(y-y_s) + \alpha_2(x+x_s-2h) + 2\alpha_1 h\ell \, , \qquad \text{(E-38b)}$$

respectively. The time derivatives of the variable p associated with Equations (E-38a) and (E-39b) are given by

$$\frac{dp}{dt} = - \frac{j\alpha_2(p_o)}{\sqrt{t^2-t_o{}^2}}$$

and

$$\frac{dp}{dt} = \frac{1}{|y-y_s| - p\left(\frac{x+x_s-2h}{\alpha_2{}^{(\ell)}} + \frac{2h\ell}{\alpha_1{}^{(\ell)}}\right)} \, .$$

Therefore, the first term of Equation (E-3.) may be expressed in closed form as

$$-\frac{\text{Real}[R(p_o)]}{\sqrt{t^2-t_o{}^2}}\,H(t-t_o)$$

where:

$$t_o = [(x+x_s-2h)^2 + |y-y_s|^2]^{1/2}/c \;;$$

and

$$p_o(t) = \frac{(y-y_s)t}{(y-y_s)^2 + (x+x_s-2h)^2} + j\,\frac{(x+x_s-2h)}{(y-y_s)^2 + (x+x_s-2h)^2}\,(t^2-t_o{}^2)^{\frac{1}{2}} \;.$$

Thus, a more appropriate form for the total SH-wave displacement for the delta-function displacement response is

$$\hat{u}_2(t) = \frac{1}{2\pi\rho_2 c_2{}^2}\;\frac{H(t-t_s)}{\sqrt{t^2-t_s{}^2}} - \frac{\text{Real}[R(p_o)]}{\sqrt{t^2-t_o{}^2}}\,H(t-t_o)$$

$$+\sum_{\ell=1}^{N} \text{Im}\left[(R^{-1}-R)\,\frac{R^{\ell}}{\alpha_2}\,\frac{dp}{dt}\right]_{p=p_\ell(t)} H(t-t_\ell) \;. \qquad \text{(E-39)}$$

To reduce numerical computations, the first motion approximation for the reflections may be applied to the summation term of Equation (E-39). In the first-motion approximation, only values of p near p_ℓ influence the behavior of the SH-wave displacement (Helmberger, 1968). In Equation (E-39) the only function that is not slowly varying near p_ℓ is the time derivative dp/dt. This numerical technique requires the time, t, to be expanded in a power series about the time of arrival, t_ℓ, as

$$t = t_\ell + \left(\frac{dt}{dp}\right)_{p=p_\ell}(p-p_\ell) + \frac{1}{2}\left(\frac{\partial^2 t}{\partial p^2}\right)_{p=p_\ell}(p-p_\ell)^2 + \ldots. \qquad \text{(E-40)}$$

and since $\left(\frac{dt}{dp}\right)_{p=p_\ell} = 0$,

$$t-t_\ell \cong \frac{1}{2}\left(\frac{d^2 t}{dp^2}\right)_{p=p_\ell}(p-p_\ell)^2 \;. \qquad \text{(E-41)}$$

Solving Equation (E-41) for p and taking the time derivative yields

$$\left(\frac{dp}{dt}\right) = \pm(t-t_\ell)^{-1/2} / \left(2\,\frac{\partial^2 t}{\partial p^2}\right)^{1/2}_{p=p_\ell} . \tag{E-42}$$

The second time derivative of p(at $p=p_\ell$) is obtained from Equation (E-38b) and is given by

$$\left(\frac{\partial^2 t}{\partial p^2}\right)_{p=p_\ell} = -\left(\frac{x+x_s-2h}{\alpha_2{}^3 c_2{}^2} + \frac{2h\ell}{\alpha_1{}^3 c_1{}^2}\right) . \tag{E-43}$$

Substituting Equation (E-42) into Equation (E-39) yields the first-motion approximation for reflections in the surface layer. The total SH-wave displacement for the delta-function response at the detector location is, therefore

$$\hat{u}_2(t) = \frac{1}{2\pi\rho_2 c_2{}^2} \quad \frac{H(t-t_s)}{\sqrt{t^2-t_s{}^2}} - \frac{\text{Real}[R(p_o)]}{\sqrt{t^2-t_o{}^2}}\, H(t-t_o)$$

$$+ \sum_{\ell=1}^{N} \left[(R^{-1} - R)\,\frac{R^\ell}{\alpha_2}\right]_{p=p_\ell} \frac{1}{(2S_\ell)^{1/2}} \; \frac{1}{\sqrt{(t-t_\ell)}} \quad , \tag{E-44}$$

where:

$$S_\ell = \frac{x+x_s-2h}{\alpha_2{}^3 c_2{}^2} + \frac{2h\ell}{\alpha_1{}^3 c_1{}^2}$$

and

$$p_\ell = \frac{\sin\,\theta_2{}^{(\ell)}}{c_2} .$$

The first term of Equation (E-44) corresponds to the direct SH-wave pulse having an arrival time, t_s, while the second term represents the SH-wave pulse reflected in the bedrock-surface layer interface having an arrival time, t_o. The summation term is associated with multiple reflections of SH-wave pulses in the bedrock and in the surface layer having arrival times, t_ℓ (for ℓ = 1,2, 3,... N).

For a transient SH-wave pulse acting along the z-axis parallel to the cylindrical cavity and consisting of a body force, f(t), the transient SH-wave displacement is obtained by convolution; that is,

$$\hat{u}_2^{(i)}(t) = \frac{1}{2\pi\rho_2 c_2^2} f(t) * \hat{T}_2^{(i)}(t) \qquad (E-45)$$

and its time derivative is

$$\frac{d\hat{u}_2^{(i)}(t)}{dt} = \frac{1}{2\pi\rho_2 c_2^2} \left\{\frac{d}{dt}\left(f(t) * \hat{T}_2^{(i)}(t)\right)\right\} \qquad (E-46)$$

where:

$$\hat{T}_2^{(i)}(t) = \frac{H(t-t_s)}{\sqrt{t^2-t_s^2}} - \frac{Real[R(p_o)]}{\sqrt{t^2-t_o^2}} H(t-t_o)$$

$$+ \sum_{\ell=1}^{N} \left\{(R^{-1} - R) \frac{R^\ell}{\alpha_2}\right\}_{p=p_\ell} \frac{1}{(2S_\ell)^{1/2}} \frac{1}{\sqrt{(t-t_\ell)}} \qquad (E-47)$$

in which:

$$S_\ell = \frac{x+x_s-2h}{\alpha_2{}^3 c_2{}^2} + \frac{2h\ell}{\alpha_1{}^3 c_1{}^2}$$

and

$$p_\ell = \frac{\sin\theta_2^{(\ell)}}{c_2} .$$

E.2.2 Detector in the Surface Layer

The SH-wave displacement for the line source in the bedrock layer and the detector in the surface layer as given by Equation (E-9) may be expressed in terms of two integrals

$$u_1 = \frac{1}{2\pi\rho_2 c_2^2} \sum_{\ell=0}^{N} \operatorname{Im}\int_C \frac{T(p)R(p)^\ell}{\alpha_2} dp$$

$$\times e^{-S\{p(y-y_s) + \alpha_1[x+h(2\ell+1) + \alpha_2(x_s-h)\}}$$

$$+ \frac{1}{2\pi\rho_2 c_2^2} \sum_{\ell=0}^{N} \operatorname{Im}\int_C \frac{T(p)R(p)^\ell}{\alpha_2} dp$$

$$\times e^{-S\{p(y-y_s) + \alpha_1[-x+h(2\ell+1) + \alpha_2(x_s-h)\}} . \qquad (E-48)$$

From these expressions, the SH-wave displacement at the surface (x = 0) is

$$u_1 = \frac{1}{2\pi\rho_2 c_2^2}\, 2\sum_{\ell=0}^{N} \mathrm{Im}\int_C \frac{T(p)R(p)^\ell}{\alpha_2} e^{-S[p(y-y_s) + \alpha_1 h(2\ell+1) + \alpha_2(x_s-h)]} dp \qquad \text{(E-49)}$$

and the generalized reflection displacement is given by

$$u_1(S) = \frac{1}{2\pi\rho_2 c_2^2}\, 2\sum_{\ell=0}^{N} \mathrm{Im}\int_{t_\ell}^{\infty} \left[\frac{T(p)R(p)^\ell}{\alpha_2}\,\frac{dp}{dt}\right]_{p=p_\ell(t)} e^{-st}\, dt \;. \qquad \text{(E-50)}$$

In the time domain, the generalized reflections for the delta-function displacement response can be identified directly from Equation (E-50) as

$$\hat{u}_1(t) = \frac{1}{2\pi\rho_2 c_2^2}\, 2\sum_{\ell=0}^{\infty} \mathrm{Im}\left[\frac{R^\ell(p)}{\alpha_2}\, T(p)\, \frac{dp}{dt}\right]_{p=p_\ell(t)} H(t-t_\ell) \;, \qquad \text{(E-51)}$$

where $p_\ell(t)$ is a solution of the nonlinear equation given by

$$t = p(y-y_s) + \alpha_2(x_s-h) + \alpha_1 h(2\ell+1) \;. \qquad \text{(E-52)}$$

The time derivative of the variable, p, associated with Equation (E-52) is

$$\frac{dP}{dt} = \frac{1}{|y-y_s| - p\left[\frac{x_s-h}{\alpha_2} + \frac{h(2\ell+1)}{\alpha_1}\right]} \qquad \text{(E-53)}$$

in which

$$\alpha_j = \left(\frac{1}{c_j^2} - p_\ell^2\right)^{1/2} \;;\ \text{for } j = 1,\ 2 \;.$$

The equation to compute the times of arrival of the reflections for a detector at the surface is given by

$$t_\ell = \frac{x_s - h}{c_2 \cos\theta_2^{(\ell)}} + \frac{h(2\ell+1)}{c_1 \cos\theta_1^{(\ell)}} \qquad \text{(E-54)}$$

where the angles θ_1 and θ_2 are the angles of the ray trajectories in the surface layer and in the semi-infinite half-space, respectively. The ray contribution associated with Equation (E-54) is shown in Figure E-3. The horizontal distance between source and detector can be inferred directly from the ray geometry shown in Figure E-3 as

$$r_\ell = (x_s-h) \tan \theta_2^{(\ell)} + h(2\ell+1) \tan \theta_1^{(\ell)} \; .$$

The angles $\theta_1^{(\ell)}$ and $\theta_2^{(\ell)}$ may be obtained for an arbitrary point of observation p(x,y) by solving the following system of equations:

$$r_\ell = (x_s-h) \tan \theta_2^{(\ell)} + h(2\ell+1) \tan \theta_1^{(\ell)} \qquad \text{(E-55)}$$

$$\frac{\sin \theta_2^{(\ell)}}{c_2} = \frac{\sin \theta_1^{(\ell)}}{c_1} = p \quad .$$

These equations may be expressed in terms of the ray-path parameter, p, as a nonlinear equation given by

$$r_\ell = \frac{(x_s-h)p}{\sqrt{c_2^{-2} - p^2}} + \frac{h(2\ell+1)p}{\sqrt{c_1^{-2} - p^2}} \qquad \text{(E-56)}$$

$$\text{for } 0 < p < 1/c_2 \; .$$

For a prescribed source-detector position, Equation (E-56) must be solved numerically for the ray parameter, p. In this way the pair of angles of incidence $[\theta_1^{(\ell)}, \theta_2^{(\ell)}]$ may be determined and substituted in Equation (E-54) to obtain the arrival times, t_ℓ, for $\ell = 0,1,2,\ldots N$.

Implementing the first-motion approximation in Equation (E-51), the SH-wave displacement, $\hat{u}_1(t)$, becomes

$$\hat{u}_1(t) \cong \frac{1}{2\pi\rho_2 c_2^2} \sum_{\ell=1}^{N} \frac{2}{\sqrt{(2S_\ell)}} \, \mathrm{Re}\left[\frac{R^\ell(p_\ell)T(p_\ell)}{\alpha_2(p_\ell)}\right] \frac{1}{\sqrt{(t-t_\ell)}} , \qquad \text{(E-57)}$$

where:

$$S_\ell = \frac{(x_s-h)}{\alpha_2^3 c_2^2} + \frac{h(2\ell+1)}{\alpha_1^3 c_1^2}$$

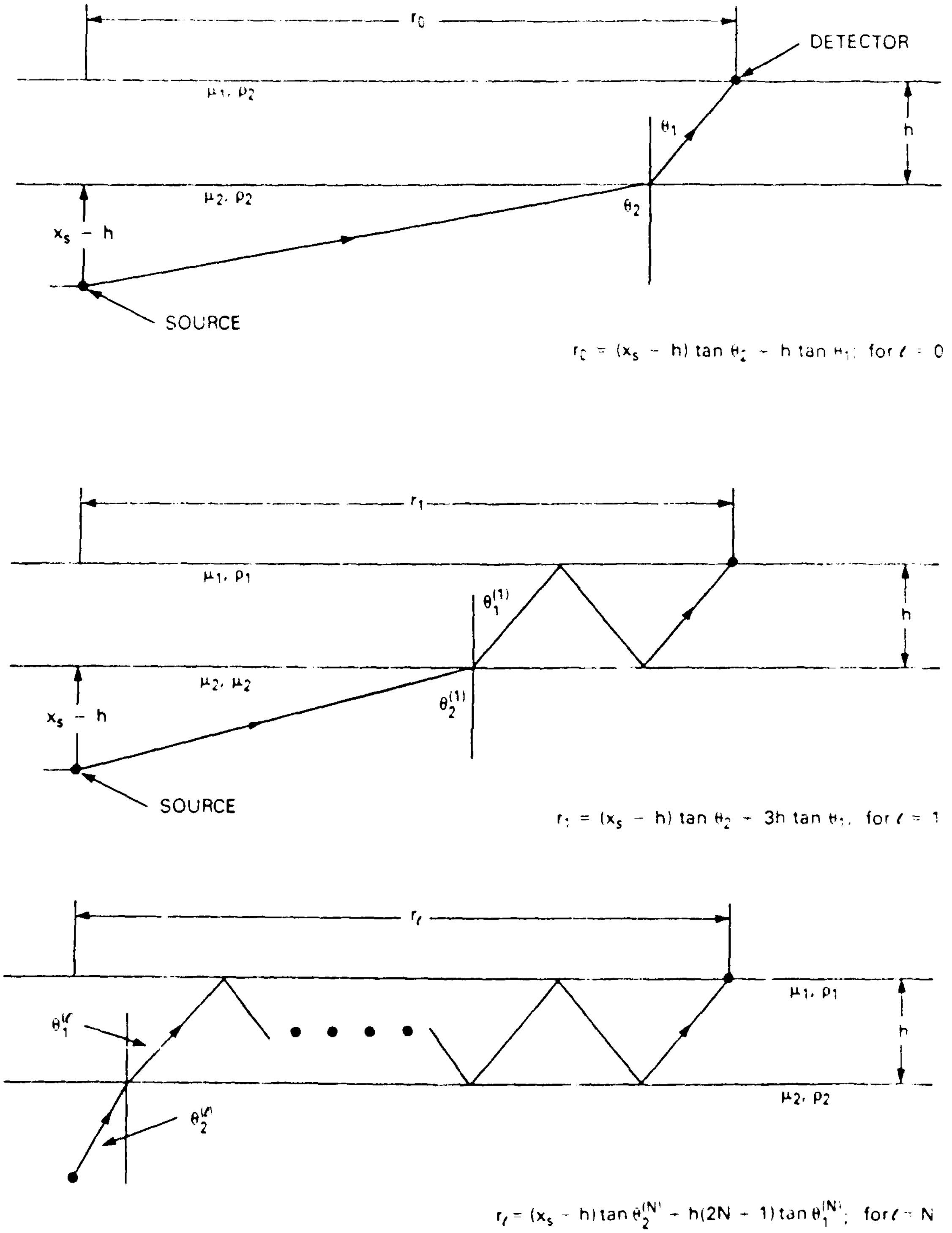


FIGURE E-3. GEOMETRY OF SH-WAVE RAY PATHS FROM A LINE SOURCE IN THE BEDROCK LAYER TO A DETECTOR AT THE SURFACE SHOWING THE CONTRIBUTIONS ASSOCIATED WITH THE POWER SERIES EXPANSION OF EQUATION (E-51)

and

$$p_\ell = \frac{\sin \theta_2^{(\ell)}}{c_2} .$$

For a transient SH-wave pulse acting along the z-axis parallel to the cylindrical cavity and consisting of a body force, f(t), the transient SH-wave displacement is obtained by convolution; that is,

$$\hat{u}_1(t) = \frac{1}{2\pi\rho_2 c_2{}^2} \left(f(t) * \hat{T}_1{}^{(i)}(t) \right) \qquad \text{(E-58)}$$

and its time derivative is

$$\frac{d\hat{u}_1(t)}{dt} = \frac{1}{2\pi\rho_2 c_2{}^2} \frac{d}{dt} \left(f(t) * \hat{T}_1{}^{(i)}(t) \right) \qquad \text{(E-59)}$$

where:

$$\hat{T}_1{}^{(i)} = \sqrt{2} \sum_{\ell=0}^{N} \frac{1}{\sqrt{S_\ell}} \operatorname{Re}\left[\frac{R^\ell(p)T(p)}{\alpha_2(p)}\right]_{p=p_\ell} \frac{1}{\sqrt{(t-t_o)}} . \qquad \text{(E-60)}$$

APPENDIX F

ASYMPTOTIC SOLUTION OF THE INTEGRAL Q_{nm}

APPENDIX F

Asymptotic Solution of the Integral Q_{nm}

Making the substitution $\lambda = k_2 \sin \theta$ in the integral Q_{nm} given by Equation (110a) leads to

$$Q_{nm} = (-j)^{n+m} \frac{1}{\pi} \int_{\Gamma} \varepsilon_1(\theta) e^{jk_2(2H-h-x)\cos\theta - j\theta(n+m)} d\theta$$

$$\times \left[e^{jk_2\alpha(\theta)(x+h)} + e^{jk_2\alpha(\theta)(h-x)} \right] \qquad \text{(F-1)}$$

where the integration path Γ is shown in Figure (D-1) and $\varepsilon_1(\theta)$ is given by

$$\varepsilon_1(\theta) = \frac{T(\theta)}{1 - R(\theta) e^{j2k_2 h\alpha(\theta)}} , \qquad \text{(F-2)}$$

in which

$$R(\theta) = \frac{\zeta^2\alpha(\theta) - \cos\theta}{\zeta^2\alpha(\theta) + \cos\theta} ;$$

$$T(\theta) = \frac{2 \cos\theta}{\zeta^2\alpha(\theta) + \cos\theta} ;$$

$$\alpha(\theta) = (\nu^2 - \sin^2\theta)^{1/2} ;$$

$$\zeta^2 = \mu_1/\mu_2 ;$$

and

$$\nu = k_1/k_2 .$$

As was done in Appendix D, the asymptotic solution of the integral Q_{nm} may be derived using the saddle-point method. To obtain an appropriate form of the integral given in Equation (F-1) and to remove the singularities, the integrand, $\varepsilon_1(\theta)$, may be expanded in the power series

$$\varepsilon_1(\theta) = T(\theta) \sum_{\ell=0}^{\infty} R^{\ell}(\theta) e^{2jk_2 h\alpha\ell} \qquad \text{(F-3)}$$

where:

$$\alpha \equiv \alpha(\theta) = (\nu^2 - \sin^2\theta)^{1/2} .$$

Equation (F-1) may be express as

$$Q_{nm} = Q_{nm}^{+} + Q_{nm}^{-} \qquad (F-4)$$

where:

$$Q_{nm}^{+} = (-j)^{n+m} \frac{1}{\pi} \sum_{\ell=0}^{\infty} \int_{\Gamma} T(\theta)R(\theta)^{\ell} e^{jk_2(2H-h-x)[g^{+}(\theta) + \cos\theta]} d\theta e^{j\theta(n+m)} \qquad (F-5)$$

and

$$Q_{nm}^{-} = (-j)^{n+m} \frac{1}{\pi} \sum_{\ell=0}^{\infty} \int_{\Gamma} T(\theta)R(\theta)^{\ell} e^{jk_2(2H-h-x)[g^{-}(\theta) + \cos\theta]} d\theta e^{j\theta(n+m)} \qquad (F-6)$$

in which

$$g^{+}(\theta) = \frac{\alpha(x+h+2\ell h)}{2H-h-x}$$

and

$$g^{-}(\theta) = \frac{\alpha(h-x+2\ell h)}{2H-h-x} .$$

For a detector located at the surface (x = 0)

$$g(\theta) \equiv g^{+}(\theta) = g^{-}(\theta) = \frac{\alpha(1+2\ell)h}{2H-h} \;;\; x \to 0 \qquad (F-7)$$

and the integral Q_{nm} is reduced to

$$Q_{nm} = (-j)^{n+m} \frac{2}{\pi} \sum_{\ell=0}^{\infty} \int_{\Gamma} T(\theta)R^{\ell} e^{jk_2(2H-h)[g(\theta) + \cos\theta]} d\theta e^{j\theta(n+m)} . \qquad (F-8)$$

To solve the integral Q_{nm} using the saddle point method, the following integral is considered.

$$I_\ell = \int_\Gamma T(\theta)R(\theta)^\ell e^{jk_2(2H-h)(g(\theta) + \cos\theta) + j\theta(n+m)} d\theta \qquad (F-9)$$

which has the general form given by

$$I_\ell = \int_\Gamma F(\theta)e^{\eta f(\theta)} d\theta \qquad (F-10)$$

where:

$$\eta = k_2(2H-h) ;$$

$$f(\theta) = j\left(g(\theta) + \cos\theta\right) = j\left(\cos\theta + e_\ell\alpha\right) ;$$

$$F(\theta) = T(\theta)R^\ell(\theta)e^{j\theta(n+m)} ;$$

and

$$e_\ell = \frac{(2\ell+1)h}{2H-h} .$$

The saddle point, θ_o, is determined from

$$\frac{\partial}{\partial\theta}\left(\cos\theta + e_\ell\alpha(\theta)\right) = 0$$

which gives

$$\theta = n\pi; \text{ for } n = 0,\pm 1,\pm 2,\ldots$$

Since the saddle point must be within the interval $-\pi/2 < \theta_o < \pi/2$, then θ_o must be equal to zero. Equation (F-10), as solved by the saddle-point method, has the form

$$I_\ell(\eta) = F(\theta_o)e^{\eta f(\theta_o)}\sqrt{\frac{2\pi}{-\eta f''}}\ \Big\{1 + \frac{1}{2\eta f''}\Big[\frac{f'''}{f''}\frac{F'}{F}$$

$$+ \frac{1}{4}\frac{f^{iv}}{f''} - \frac{5}{12}\frac{(f''')^2}{(f'')^2} - \frac{F''}{F}\Big] + \ldots\Big\} . \qquad (F-11)$$

The functions F and f and their derivatives are evaluated at $\theta_o = 0$. These functions are given by

$$\begin{gathered}
F(0) = \frac{2}{\zeta^2\nu + 1}\left(\frac{\zeta^2\nu - 1}{\zeta^2\nu + 1}\right)^{\ell} \\
\left(\frac{F''}{F}\right)_{\theta_o=0} = \left\{\frac{2\zeta^2\nu\ell}{(1 - \zeta^2\nu)(1 + \zeta^2\nu)} - \frac{\zeta^2(\nu - 1/\nu)}{1 + \zeta^2\nu} - (n+m)^2\right\} \\
f(0) = j(1 + e_{\ell}\nu) \\
f''(0) = -j(1 + e_{\ell}\nu)
\end{gathered} \tag{F-12}$$

and

$$\left(\frac{f^{iv}}{f''}\right)_{\theta_o=0} = - \frac{1 + \frac{4}{\nu}\, e_{\ell}(1 - 3/4\nu^2)}{1 + e_{\ell}/\nu} .$$

Thus, the asymptotic solution of the integral I_{ℓ} is

$$\begin{gathered}
I_{\ell} = T_o R_o^{\ell} b_{\ell} \sqrt{\frac{2\pi}{k_2(2H-h) + k_2(2\ell+1)h\nu}}\; e^{jk_2(2H-h) + jk_2(2\ell+1)h\nu - \frac{\pi}{4} j} \\
\times \left[1 + \frac{j}{2k_2(2H-h) + 2k_2 \frac{(2\ell+1)h}{\nu}}\right]\left[\frac{1}{4} T_1 + T_2\right]
\end{gathered} \tag{F-13}$$

where:

$$T_o = \frac{2}{\zeta^2\nu + 1} ;$$

$$R_o = \frac{\zeta^2\nu - 1}{\zeta^2\nu + 1} ;$$

$$b_{\ell} = \left[\frac{1 + (2\ell+1)\nu h/(2H-h)}{1 + (2\ell+1)\frac{h}{\nu}/(2H-h)}\right]^{\frac{1}{2}} ;$$

$$T_1 = \left(\frac{f^{iv}}{f''}\right)_{\theta_o=0} \quad ;$$

and

$$T_2 = \left(\frac{F''}{F}\right)_{\theta_o=0} \quad .$$

Finally, replacing the asymptotic solution of the integral I_ℓ in Equation (F-6) leads to the asymptotic solution of the integral, Q_{nm}, at x = 0. That is,

$$Q_{nm} \cong 2 \sum_{\ell=0}^{N} b_\ell T_o R_o^\ell \frac{1}{\pi} \sqrt{\frac{2\pi}{k_2(2H-h) + k_2(2\ell+1)h\nu}} \, e^{j\{k_2[(2H-h) + (2\ell+1)h\nu] - \frac{\pi}{4} - (n+m)\}}$$

$$\times \left[1 + \frac{j}{2\{k_2[2H-h+(2\ell+1)\frac{h}{\nu}]\}}\right]\left[\frac{1}{4} T_1+T_2\right] \quad . \qquad \text{(F-14)}$$

In the high-frequency limit as $2k_2(2H-h) \rightarrow \infty$, the approximate solution for the function Q_{nm} may be obtained

$$Q_{nm} \cong 2 \sum_{\ell=0}^{N} b_\ell T_o R_o^\ell H_{n+m}\{k_2[(2H-h) + (1+2\ell)h\nu]\} \qquad \text{(F-15)}$$

APPENDIX G

SYNTHETIC SEISMOGRAMS OF SH-WAVE REFLECTIONS FROM A CYLINDRICAL CAVITY IN A TWO-LAYER LOSSY HALF-SPACE

APPENDIX G

Synthetic Seismograms of SH-Wave Reflections from a Cylindrical Cavity in a Two-Layer Lossy Half-Space

The synthetic seismograms presented in this appendix highlight the important case in which the medium is represented as a two-layer medium containing a cylindrical cavity target in the lower (bedrock) medium. The surface layer represents the weathered layer and is typified in the model as a relatively low velocity unconsolidated soil. The constitutive parameters of SH-wave velocities and the mass densities of each layer are maintained essentially the same in all of the computed cases, whereas the 2-meter diameter cavity depth, the surface layer thickness, and the absorptive characteristics (Q factor) of each layer are the variables. For convenient reference, the various parameters associated with each synthetic seismogram are tabulated in Table G-1.

TABLE G-1

LIST OF PARAMETERS USED IN LOSSY TWO-LAYER MODEL COMPUTATIONS

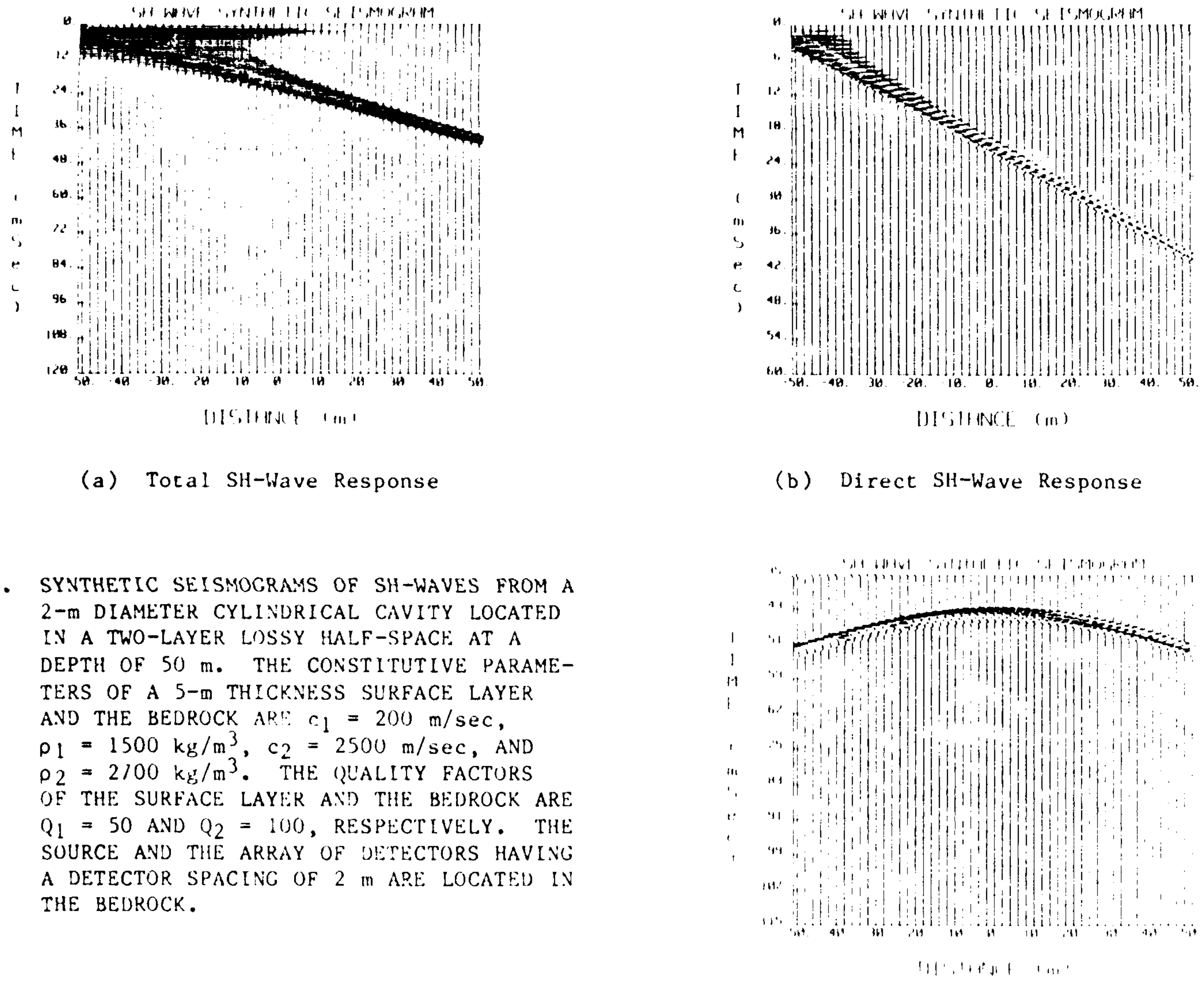
Figure No.	Cylindrical Cavity Depth (m)	Surface Layer Thickness (m)	Surface Layer Q Factor	Seismic Detector Location	Surface Layer SH Velocity (m/s)	Reflections in Surface Layer	
						Direct Wave	Cavity Reflection
G-1	50	5	50	Bedrock	200	0	1
G-2	50	10	50	Bedrock	200	0	1
G-3	50	5	10	Bedrock	200	0	1
G-4	50	10	10	Bedrock	200	0	1
G-5	100	5	50	Bedrock	200	0	1
G-6	100	10	50	Bedrock	200	0	1
G-7	100	5	10	Bedrock	200	0	1
G-8	100	10	10	Bedrock	200	0	1
G-9	50	5	50	Surface	200	0	1
G-10	50	10	50	Surface	200	0	1
G-11	50	5	10	Surface	200	0	1
G-12	50	10	10	Surface	200	0	1
G-13	100	5	50	Surface	200	0	1
G-14	100	10	50	Surface	200	0	1
G-15	100	5	10	Surface	200	0	1
G-16	100	10	10	Surface	200	0	1
G-17	50	5	50	Bedrock	500	0	1
G-18	50	5	50	Bedrock	500	1	2
G-19	50	5	50	Bedrock	500	2	3
G-20	50	5	50	Bedrock	500	3	4

TABLE G-1 (Cont'd)

Figure No.	Cylindrical Cavity Depth (m)	Surface Layer Thickness (m)	Surface Layer Q Factor	Seismic Detector Location	Surface Layer SH Velocity (m/s)	Reflections in Surface Layer	
						Direct Wave	Cavity Reflection
G-21	50	5	50	Surface	500	0	1
G-22	50	5	50	Surface	500	1	2
G-23	50	5	50	Surface	500	2	3
G-24	50	5	50	Surface	500	3	4

Invariant Parameters:

Surface Layer Mass Density = ρ_1 = 1,500 kg/m^3
Bedrock Q Factor = 100
Bedrock SH-Wave Velocity = 2,500 m/s
Bedrock Mass Density = ρ_2 = 2,700 kg/m^3
Source Located in Bedrock
Detector Separation Distance = 2 m
Cylindrical Cavity Diameter = 2 m

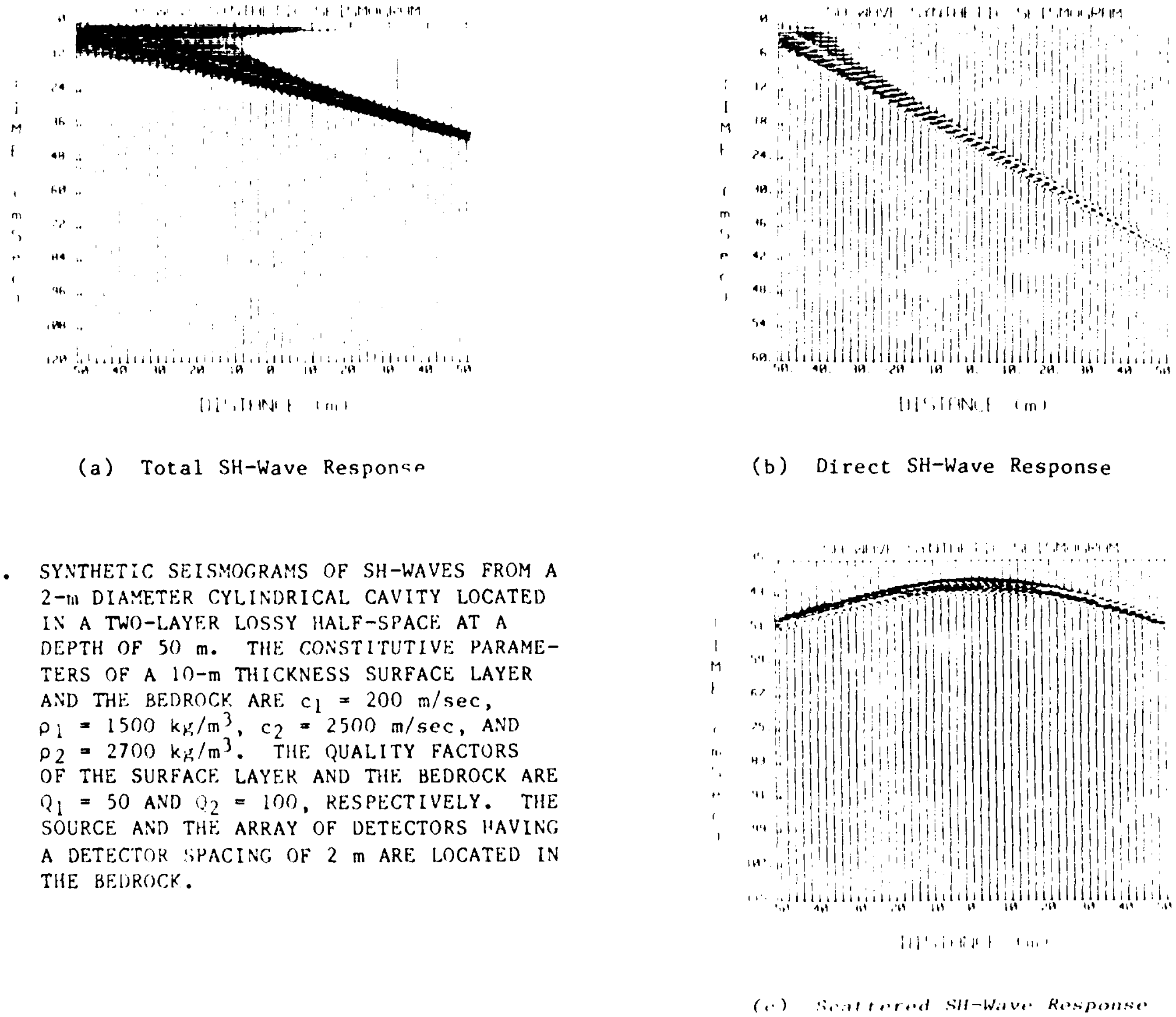


(a) Total SH-Wave Response

(b) Direct SH-Wave Response

(c) Scattered SH-Wave Response

FIGURE G-1. SYNTHETIC SEISMOGRAMS OF SH-WAVES FROM A 2-m DIAMETER CYLINDRICAL CAVITY LOCATED IN A TWO-LAYER LOSSY HALF-SPACE AT A DEPTH OF 50 m. THE CONSTITUTIVE PARAMETERS OF A 5-m THICKNESS SURFACE LAYER AND THE BEDROCK ARE c_1 = 200 m/sec, ρ_1 = 1500 kg/m^3, c_2 = 2500 m/sec, AND ρ_2 = 2700 kg/m^3. THE QUALITY FACTORS OF THE SURFACE LAYER AND THE BEDROCK ARE Q_1 = 50 AND Q_2 = 100, RESPECTIVELY. THE SOURCE AND THE ARRAY OF DETECTORS HAVING A DETECTOR SPACING OF 2 m ARE LOCATED IN THE BEDROCK.

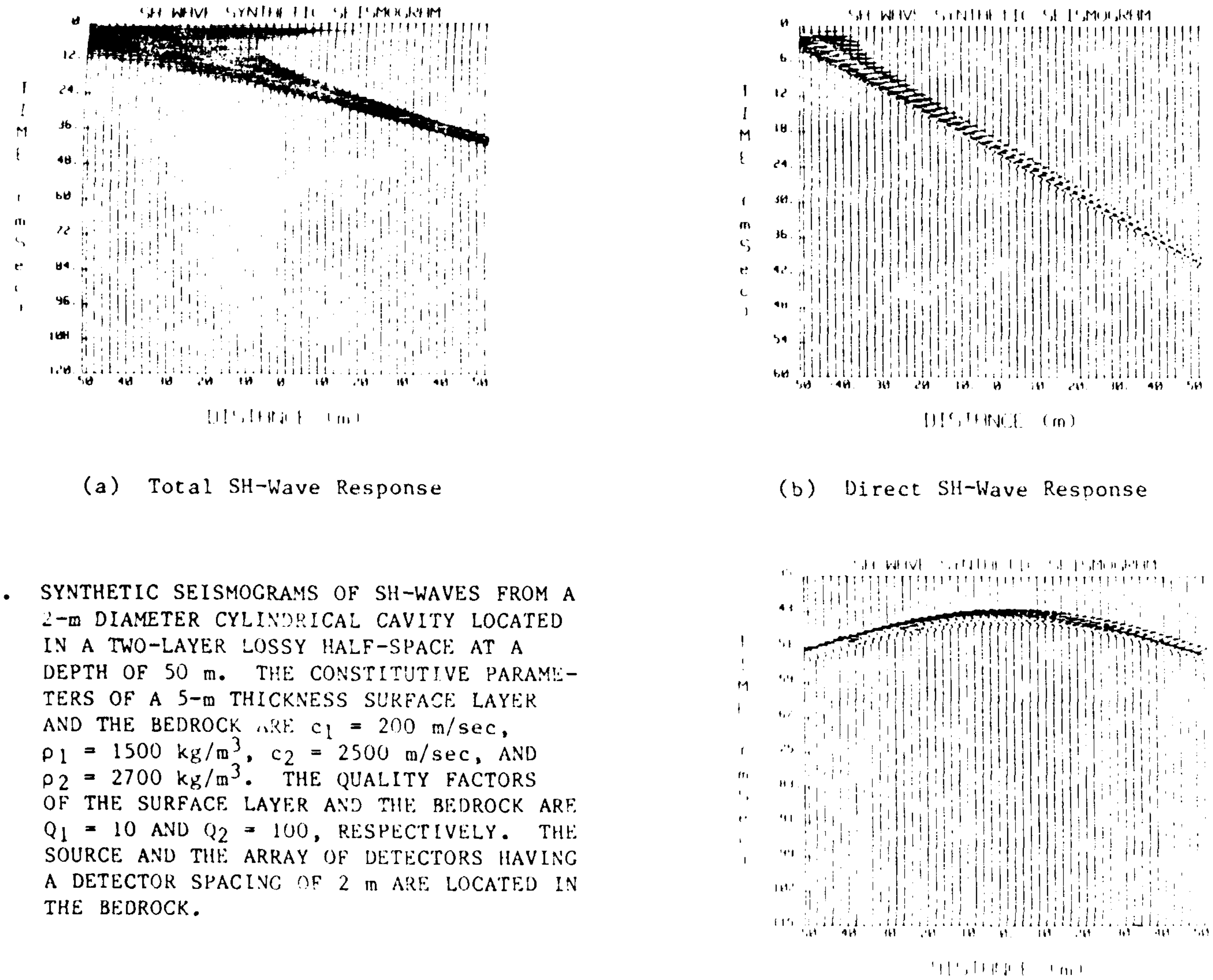


(a) Total SH-Wave Response

(b) Direct SH-Wave Response

(c) Scattered SH-Wave Response

FIGURE G-2. SYNTHETIC SEISMOGRAMS OF SH-WAVES FROM A 2-m DIAMETER CYLINDRICAL CAVITY LOCATED IN A TWO-LAYER LOSSY HALF-SPACE AT A DEPTH OF 50 m. THE CONSTITUTIVE PARAMETERS OF A 10-m THICKNESS SURFACE LAYER AND THE BEDROCK ARE $c_1 = 200$ m/sec, $\rho_1 = 1500$ kg/m^3, $c_2 = 2500$ m/sec, AND $\rho_2 = 2700$ kg/m^3. THE QUALITY FACTORS OF THE SURFACE LAYER AND THE BEDROCK ARE $Q_1 = 50$ AND $Q_2 = 100$, RESPECTIVELY. THE SOURCE AND THE ARRAY OF DETECTORS HAVING A DETECTOR SPACING OF 2 m ARE LOCATED IN THE BEDROCK.

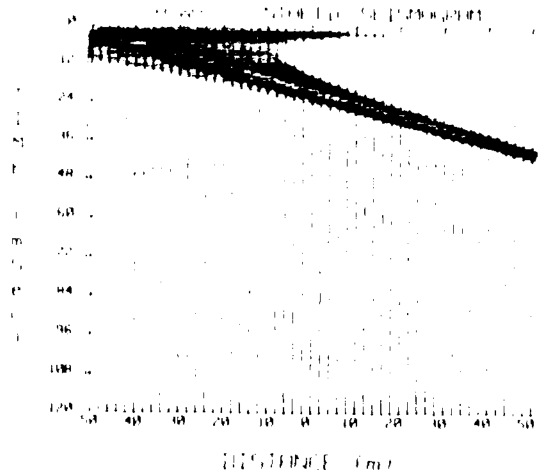


(a) Total SH-Wave Response

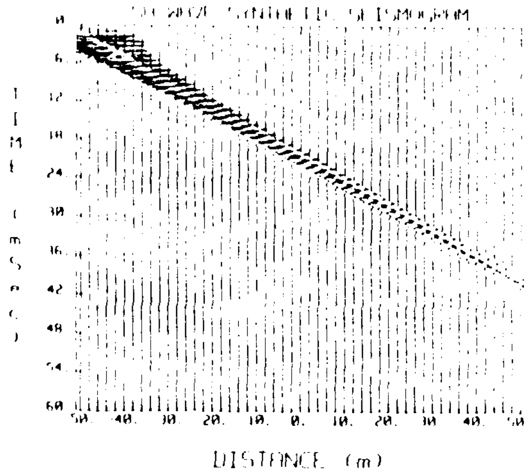
(b) Direct SH-Wave Response

(c) Scattered SH-Wave Response

FIGURE G-3. SYNTHETIC SEISMOGRAMS OF SH-WAVES FROM A 2-m DIAMETER CYLINDRICAL CAVITY LOCATED IN A TWO-LAYER LOSSY HALF-SPACE AT A DEPTH OF 50 m. THE CONSTITUTIVE PARAMETERS OF A 5-m THICKNESS SURFACE LAYER AND THE BEDROCK ARE $c_1 = 200$ m/sec, $\rho_1 = 1500$ kg/m^3, $c_2 = 2500$ m/sec, AND $\rho_2 = 2700$ kg/m^3. THE QUALITY FACTORS OF THE SURFACE LAYER AND THE BEDROCK ARE $Q_1 = 10$ AND $Q_2 = 100$, RESPECTIVELY. THE SOURCE AND THE ARRAY OF DETECTORS HAVING A DETECTOR SPACING OF 2 m ARE LOCATED IN THE BEDROCK.

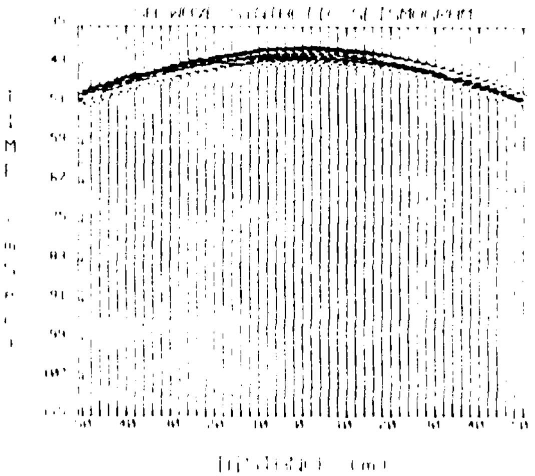


(a) Total SH-Wave Response

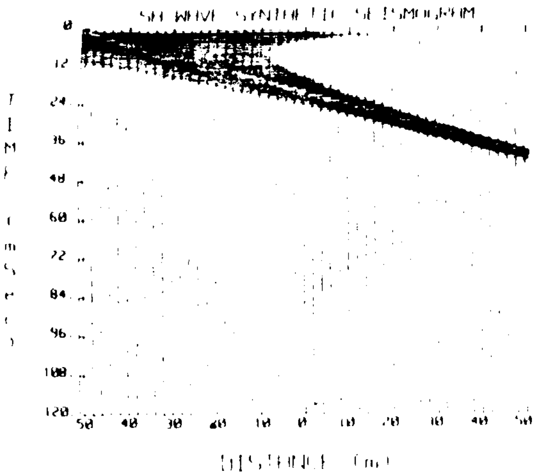


(b) Direct SH-Wave Response

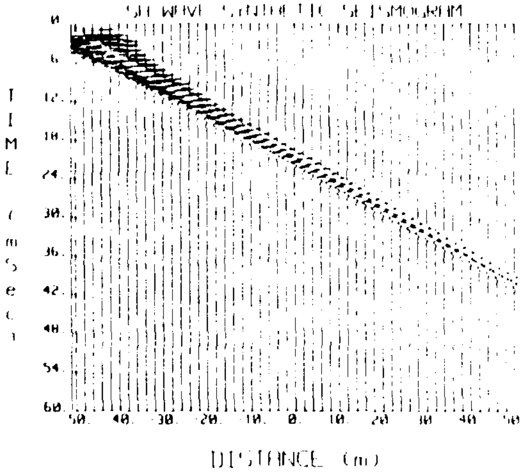
FIGURE G-4. SYNTHETIC SEISMOGRAMS OF SH-WAVES FROM A 2-m DIAMETER CYLINDRICAL CAVITY LOCATED IN A TWO-LAYER LOSSY HALF-SPACE AT A DEPTH OF 50 m. THE CONSTITUTIVE PARAMETERS OF A 10-m THICKNESS SURFACE LAYER AND THE BEDROCK ARE c_1 = 200 m/sec, ρ_1 = 1500 kg/m^3, c_2 = 2500 m/sec, AND ρ_2 = 2700 kg/m^3. THE QUALITY FACTORS OF THE SURFACE LAYER AND THE BEDROCK ARE Q_1 = 10 AND Q_2 = 100, RESPECTIVELY. THE SOURCE AND THE ARRAY OF DETECTORS HAVING A DETECTOR SPACING OF 2 m ARE LOCATED IN THE BEDROCK.



(c) Scattered SH-Wave Response

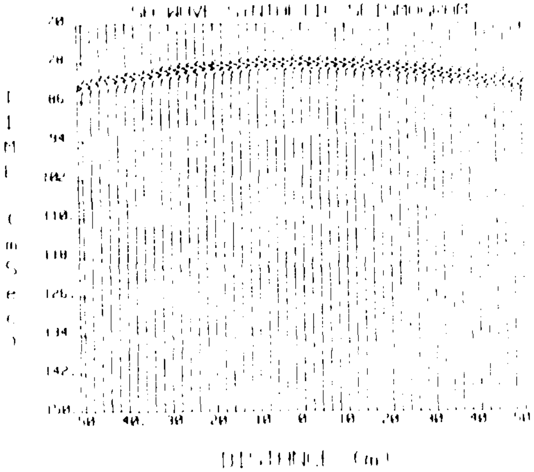


(a) Total SH-Wave Response

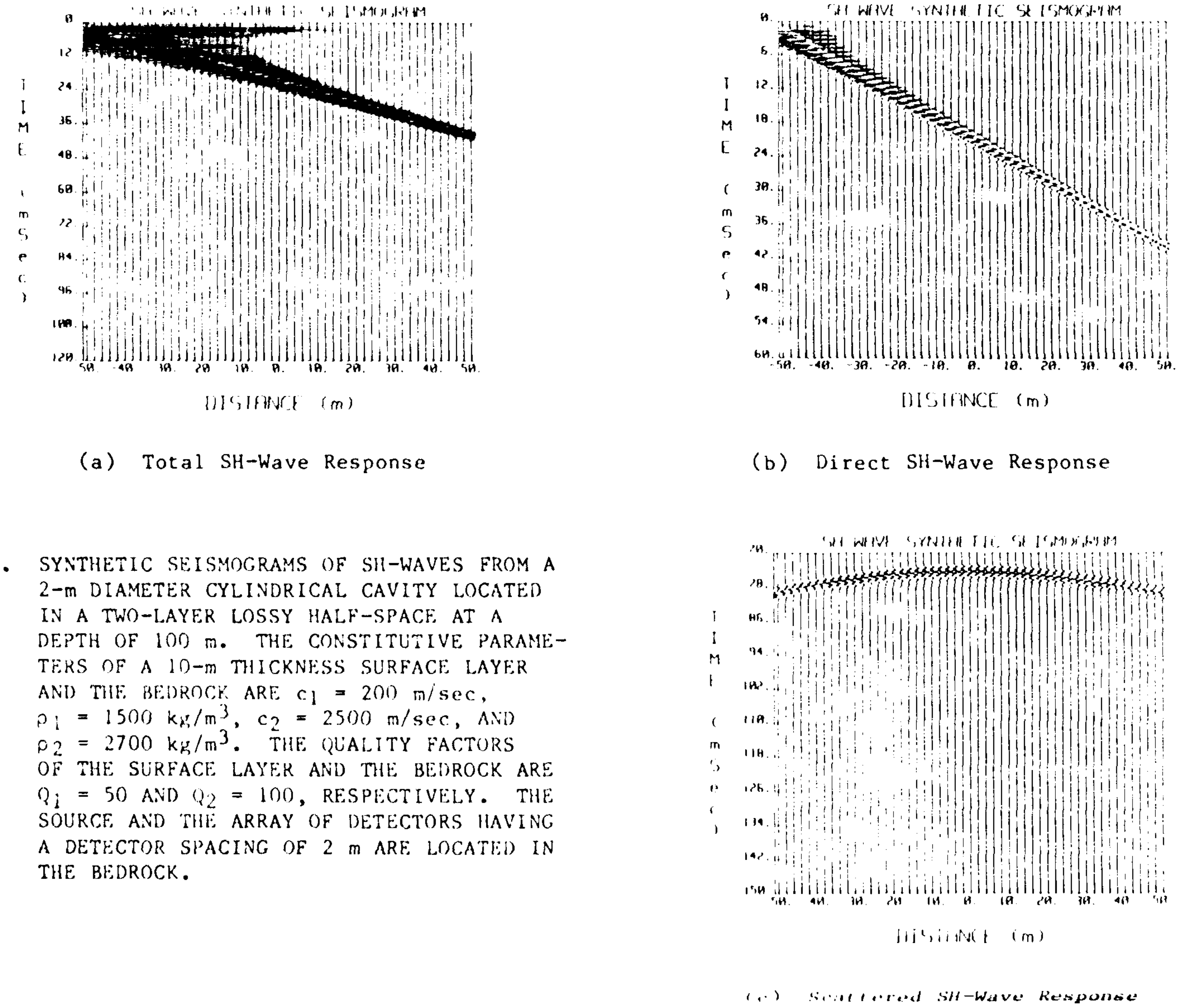


(b) Direct SH-Wave Response

FIGURE G-5. SYNTHETIC SEISMOGRAMS OF SH-WAVES FROM A 2-m DIAMETER CYLINDRICAL CAVITY LOCATED IN A TWO-LAYER LOSSY HALF-SPACE AT A DEPTH OF 100 m. THE CONSTITUTIVE PARAMETERS OF A 5-m THICKNESS SURFACE LAYER AND THE BEDROCK ARE c_1 = 200 m/sec, ρ_1 = 1500 kg/m^3, c_2 = 2500 m/sec, AND ρ_2 = 2700 kg/m^3. THE QUALITY FACTORS OF THE SURFACE LAYER AND THE BEDROCK ARE Q_1 = 50 AND Q_2 = 100, RESPECTIVELY. THE SOURCE AND THE ARRAY OF DETECTORS HAVING A DETECTOR SPACING OF 2 m ARE LOCATED IN THE BEDROCK.



(c) Scattered SH-Wave Response

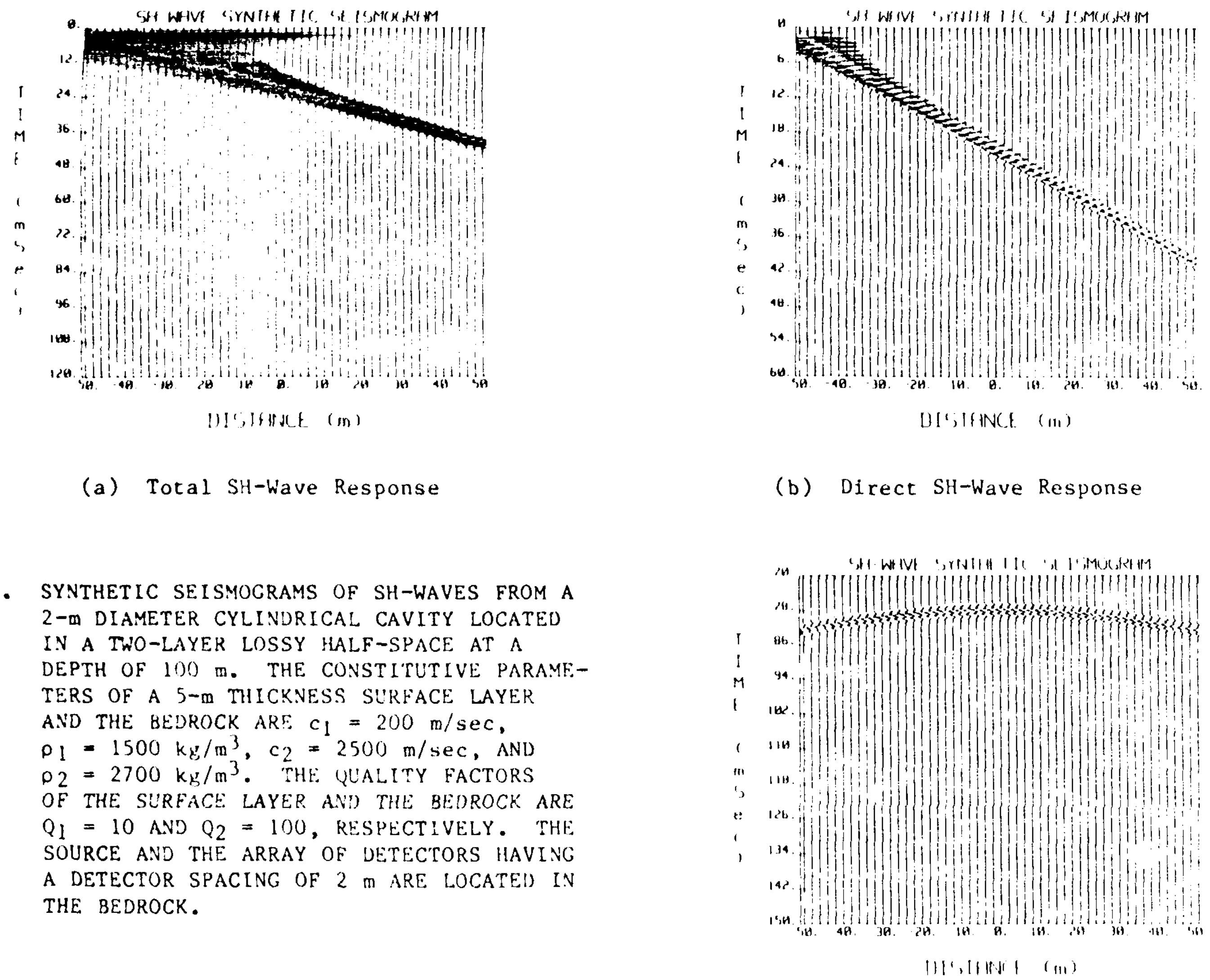


(a) Total SH-Wave Response

(b) Direct SH-Wave Response

(c) *Scattered SH-Wave Response*

FIGURE G-6. SYNTHETIC SEISMOGRAMS OF SH-WAVES FROM A 2-m DIAMETER CYLINDRICAL CAVITY LOCATED IN A TWO-LAYER LOSSY HALF-SPACE AT A DEPTH OF 100 m. THE CONSTITUTIVE PARAMETERS OF A 10-m THICKNESS SURFACE LAYER AND THE BEDROCK ARE c_1 = 200 m/sec, ρ_1 = 1500 kg/m^3, c_2 = 2500 m/sec, AND ρ_2 = 2700 kg/m^3. THE QUALITY FACTORS OF THE SURFACE LAYER AND THE BEDROCK ARE Q_1 = 50 AND Q_2 = 100, RESPECTIVELY. THE SOURCE AND THE ARRAY OF DETECTORS HAVING A DETECTOR SPACING OF 2 m ARE LOCATED IN THE BEDROCK.

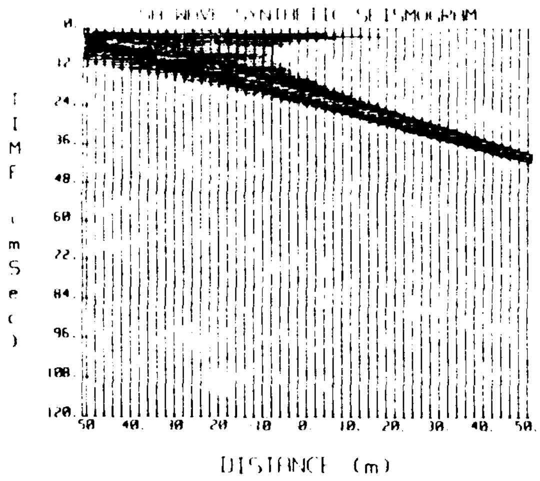


(a) Total SH-Wave Response

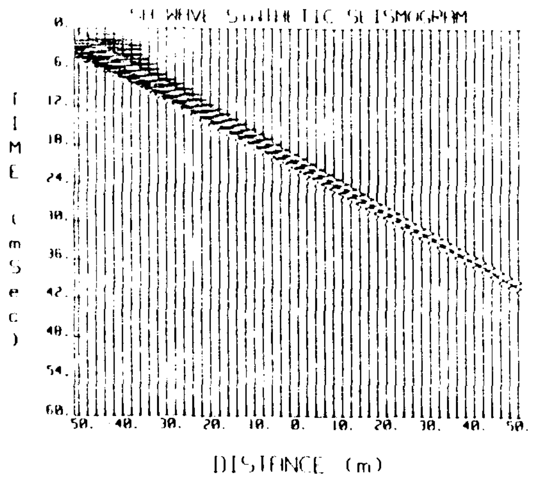
(b) Direct SH-Wave Response

(c) Scattered SH-Wave Response

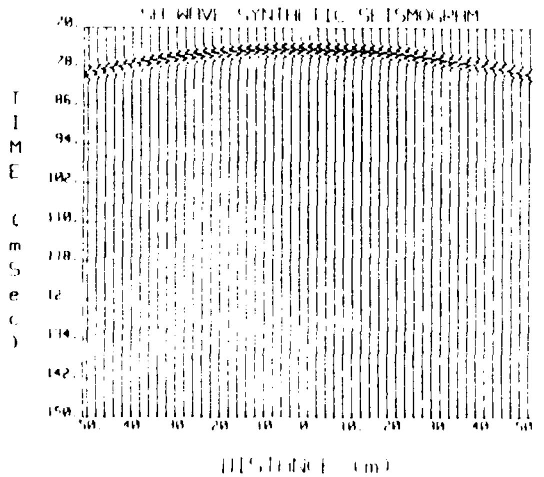
FIGURE G-7. SYNTHETIC SEISMOGRAMS OF SH-WAVES FROM A 2-m DIAMETER CYLINDRICAL CAVITY LOCATED IN A TWO-LAYER LOSSY HALF-SPACE AT A DEPTH OF 100 m. THE CONSTITUTIVE PARAMETERS OF A 5-m THICKNESS SURFACE LAYER AND THE BEDROCK ARE c_1 = 200 m/sec, ρ_1 = 1500 kg/m^3, c_2 = 2500 m/sec, AND ρ_2 = 2700 kg/m^3. THE QUALITY FACTORS OF THE SURFACE LAYER AND THE BEDROCK ARE Q_1 = 10 AND Q_2 = 100, RESPECTIVELY. THE SOURCE AND THE ARRAY OF DETECTORS HAVING A DETECTOR SPACING OF 2 m ARE LOCATED IN THE BEDROCK.



(a) Total SH-Wave Response

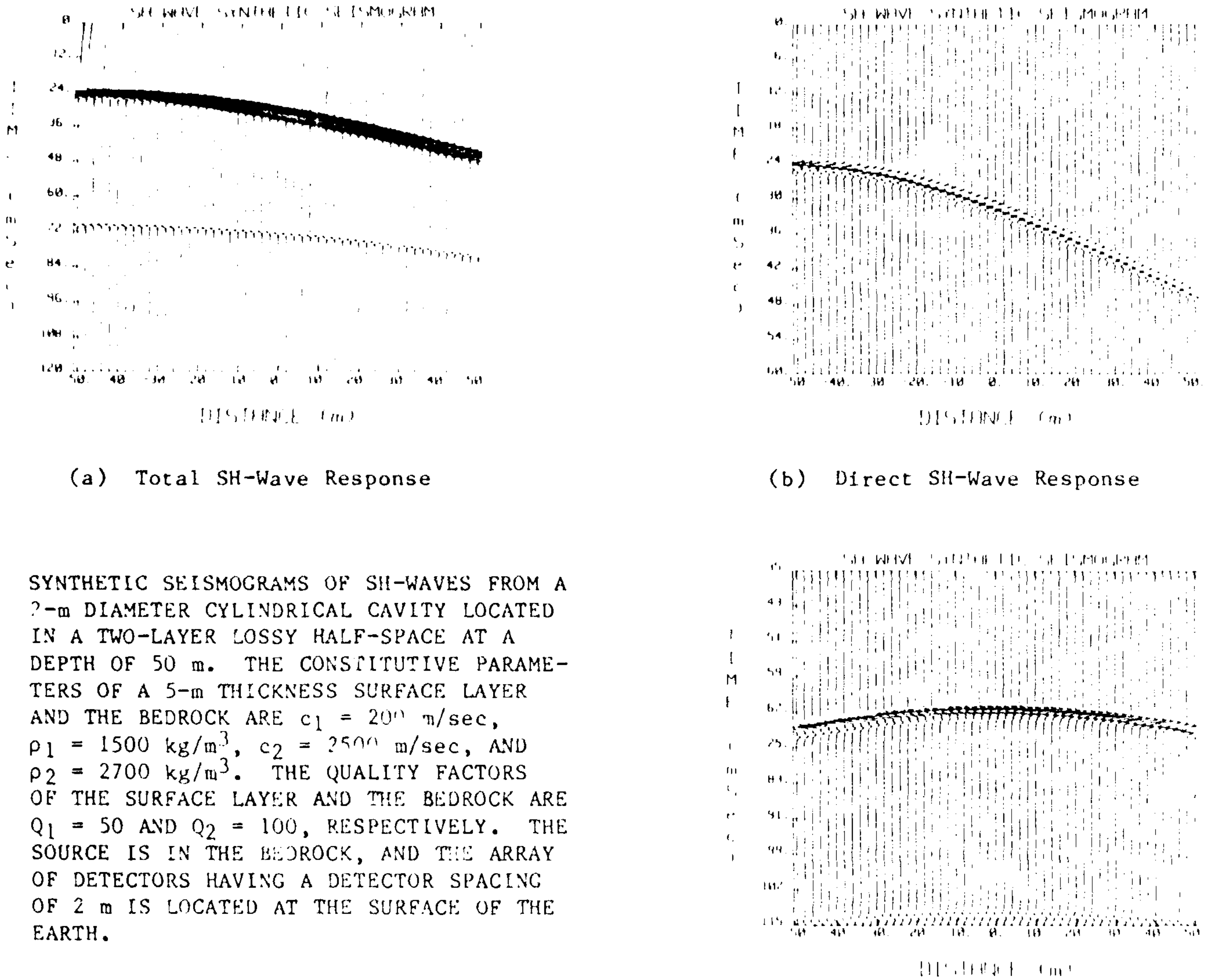


(b) Direct SH-Wave Response



(c) Scattered SH-Wave Response

FIGURE G-8. SYNTHETIC SEISMOGRAMS OF SH-WAVES FROM A 2-m DIAMETER CYLINDRICAL CAVITY LOCATED IN A TWO-LAYER LOSSY HALF-SPACE AT A DEPTH OF 100 m. THE CONSTITUTIVE PARAMETERS OF A 10-m THICKNESS SURFACE LAYER AND THE BEDROCK ARE $c_1 = 200$ m/sec, $\rho_1 = 1500$ kg/m^3, $c_2 = 2500$ m/sec, AND $\rho_2 = 2700$ kg/m^3. THE QUALITY FACTORS OF THE SURFACE LAYER AND THE BEDROCK ARE $Q_1 = 10$ AND $Q_2 = 100$, RESPECTIVELY. THE SOURCE AND THE ARRAY OF DETECTORS HAVING A DETECTOR SPACING OF 2 m ARE LOCATED IN THE BEDROCK.

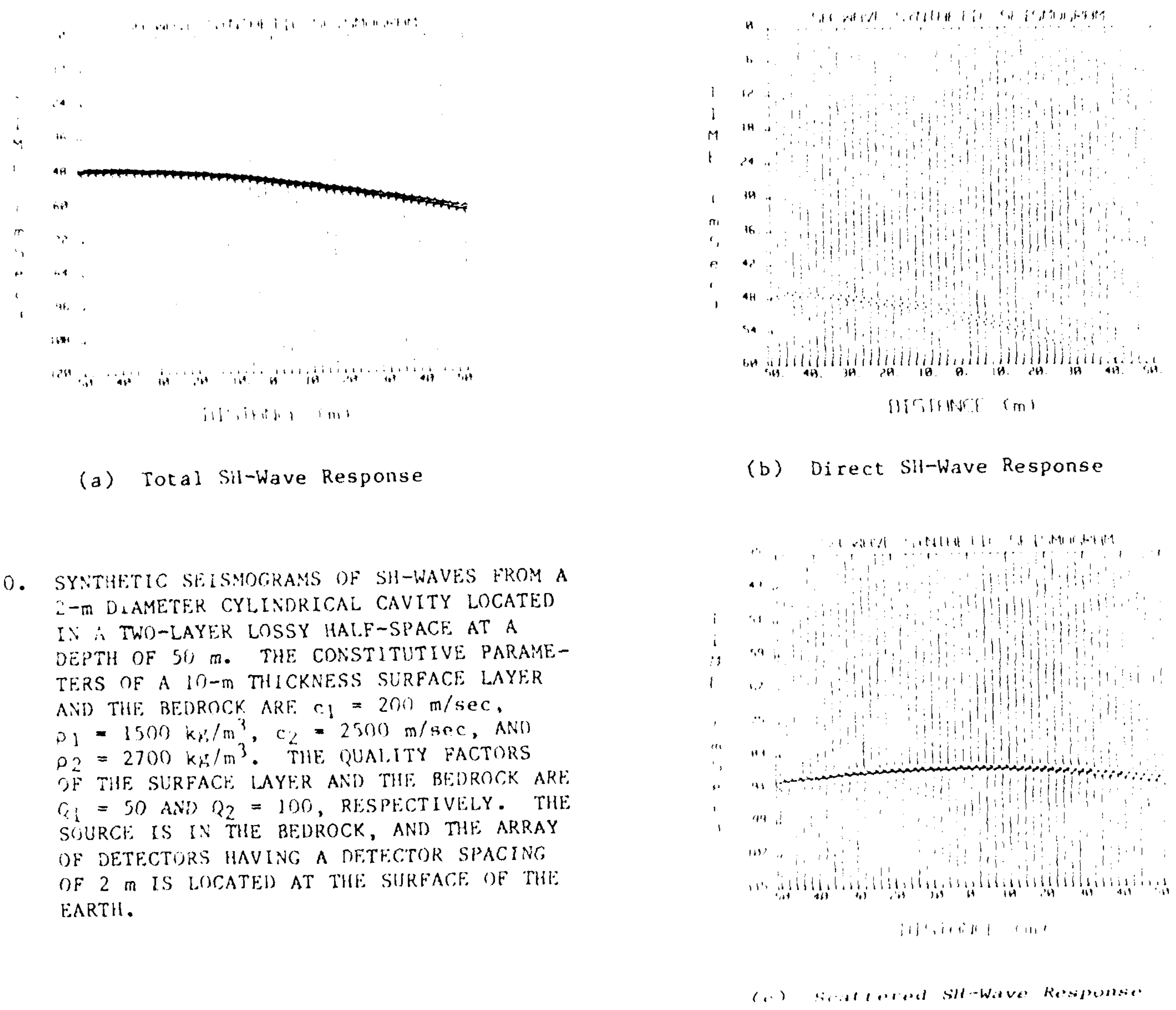


(a) Total SH-Wave Response

(b) Direct SH-Wave Response

(c) Scattered SH-Wave Response

FIGURE G-9. SYNTHETIC SEISMOGRAMS OF SH-WAVES FROM A 2-m DIAMETER CYLINDRICAL CAVITY LOCATED IN A TWO-LAYER LOSSY HALF-SPACE AT A DEPTH OF 50 m. THE CONSTITUTIVE PARAMETERS OF A 5-m THICKNESS SURFACE LAYER AND THE BEDROCK ARE c_1 = 200 m/sec, ρ_1 = 1500 kg/m^3, c_2 = 2500 m/sec, AND ρ_2 = 2700 kg/m^3. THE QUALITY FACTORS OF THE SURFACE LAYER AND THE BEDROCK ARE Q_1 = 50 AND Q_2 = 100, RESPECTIVELY. THE SOURCE IS IN THE BEDROCK, AND THE ARRAY OF DETECTORS HAVING A DETECTOR SPACING OF 2 m IS LOCATED AT THE SURFACE OF THE EARTH.

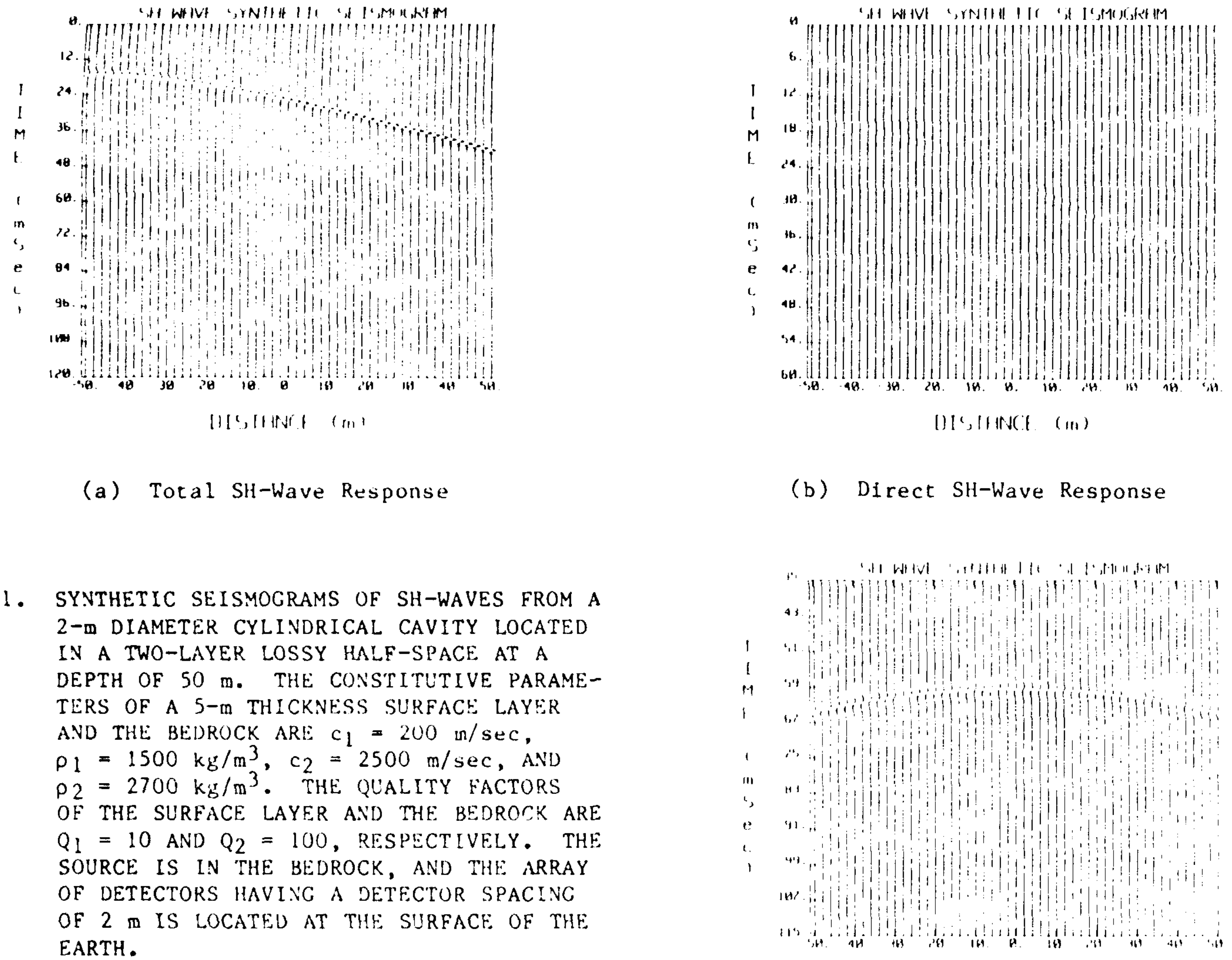


(a) Total SH-Wave Response

(b) Direct SH-Wave Response

(c) Scattered SH-Wave Response

FIGURE G-10. SYNTHETIC SEISMOGRAMS OF SH-WAVES FROM A 2-m DIAMETER CYLINDRICAL CAVITY LOCATED IN A TWO-LAYER LOSSY HALF-SPACE AT A DEPTH OF 50 m. THE CONSTITUTIVE PARAMETERS OF A 10-m THICKNESS SURFACE LAYER AND THE BEDROCK ARE c_1 = 200 m/sec, ρ_1 = 1500 kg/m^3, c_2 = 2500 m/sec, AND ρ_2 = 2700 kg/m^3. THE QUALITY FACTORS OF THE SURFACE LAYER AND THE BEDROCK ARE Q_1 = 50 AND Q_2 = 100, RESPECTIVELY. THE SOURCE IS IN THE BEDROCK, AND THE ARRAY OF DETECTORS HAVING A DETECTOR SPACING OF 2 m IS LOCATED AT THE SURFACE OF THE EARTH.

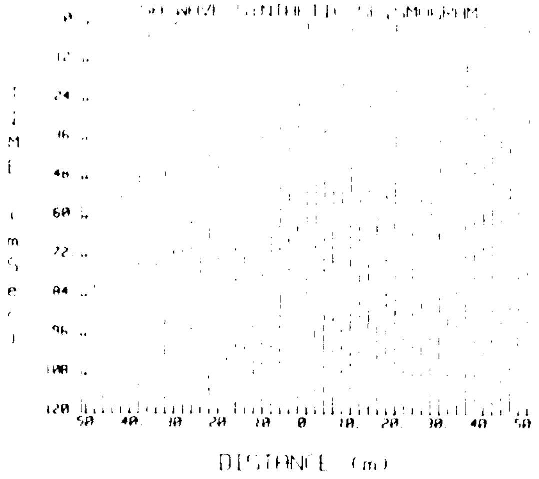


(a) Total SH-Wave Response

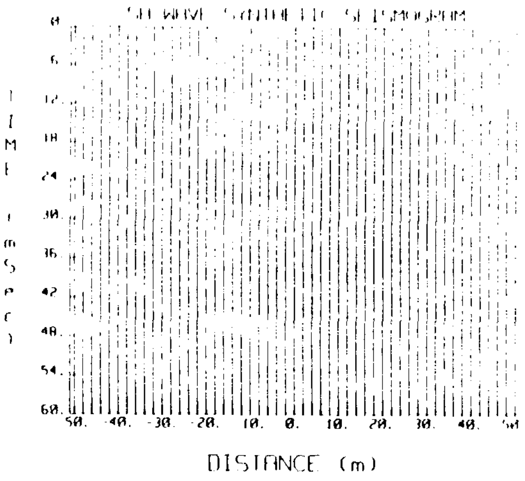
(b) Direct SH-Wave Response

(c) Scattered SH-Wave Response

FIGURE G-11. SYNTHETIC SEISMOGRAMS OF SH-WAVES FROM A 2-m DIAMETER CYLINDRICAL CAVITY LOCATED IN A TWO-LAYER LOSSY HALF-SPACE AT A DEPTH OF 50 m. THE CONSTITUTIVE PARAMETERS OF A 5-m THICKNESS SURFACE LAYER AND THE BEDROCK ARE $c_1 = 200$ m/sec, $\rho_1 = 1500$ kg/m^3, $c_2 = 2500$ m/sec, AND $\rho_2 = 2700$ kg/m^3. THE QUALITY FACTORS OF THE SURFACE LAYER AND THE BEDROCK ARE $Q_1 = 10$ AND $Q_2 = 100$, RESPECTIVELY. THE SOURCE IS IN THE BEDROCK, AND THE ARRAY OF DETECTORS HAVING A DETECTOR SPACING OF 2 m IS LOCATED AT THE SURFACE OF THE EARTH.

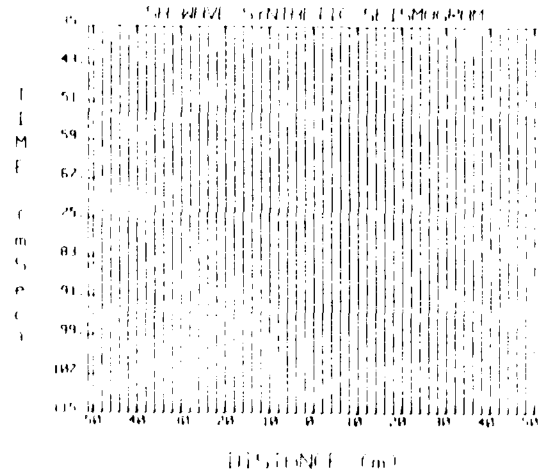


(a) Total SH-Wave Response

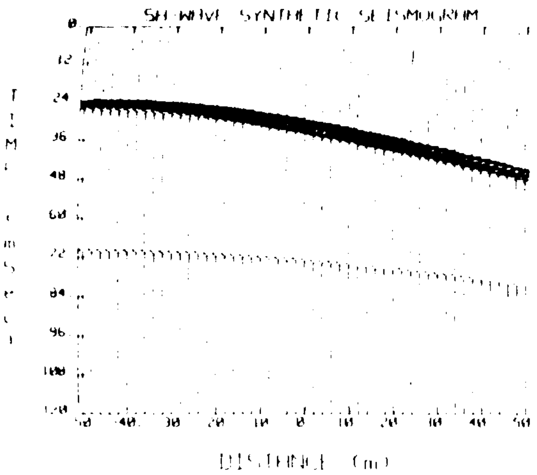


(b) Direct SH-Wave Response

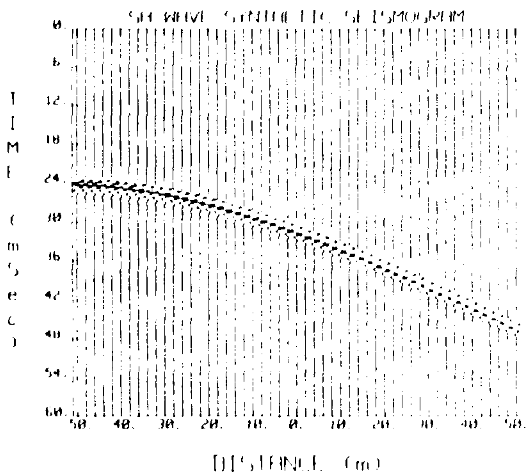
FIGURE G-12. SYNTHETIC SEISMOGRAMS OF SH-WAVES FROM A 2-m DIAMETER CYLINDRICAL CAVITY LOCATED IN A TWO-LAYER LOSSY HALF-SPACE AT A DEPTH OF 50 m. THE CONSTITUTIVE PARAMETERS OF A 10-m THICKNESS SURFACE LAYER AND THE BEDROCK ARE $c_1 = 200$ m/sec, $\rho_1 = 1500$ kg/m^3, $c_2 = 2500$ m/sec, AND $\rho_2 = 2700$ kg/m^3. THE QUALITY FACTORS OF THE SURFACE LAYER AND THE BEDROCK ARE $Q_1 = 10$ AND $Q_2 = 100$, RESPECTIVELY. THE SOURCE IS IN THE BEDROCK, AND THE ARRAY OF DETECTORS HAVING A DETECTOR SPACING OF 2 m IS LOCATED AT THE SURFACE OF THE EARTH.



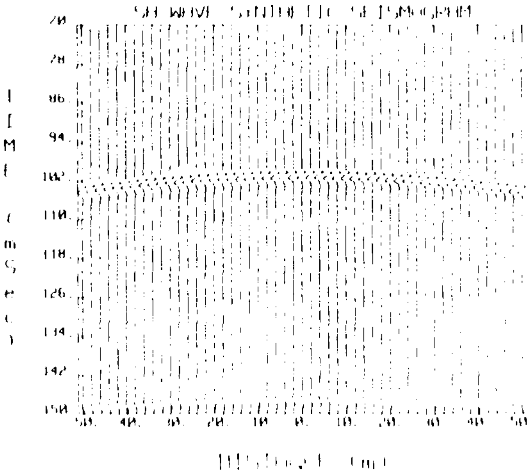
(c) Scattered SH-Wave Response



(a) Total SH-Wave Response

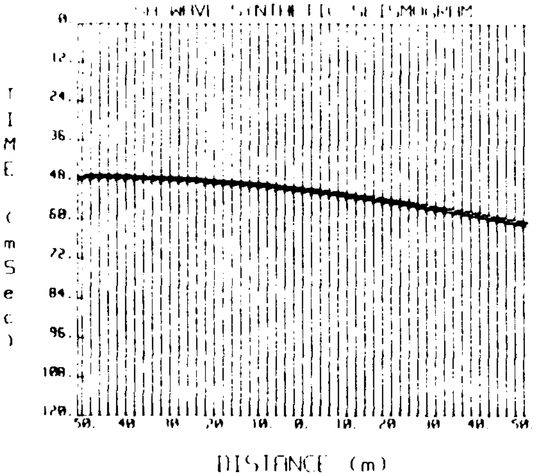


(b) Direct SH-Wave Response

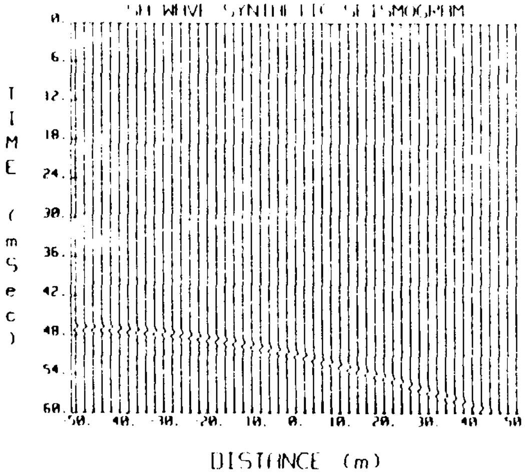


(c) Scattered SH-Wave Response

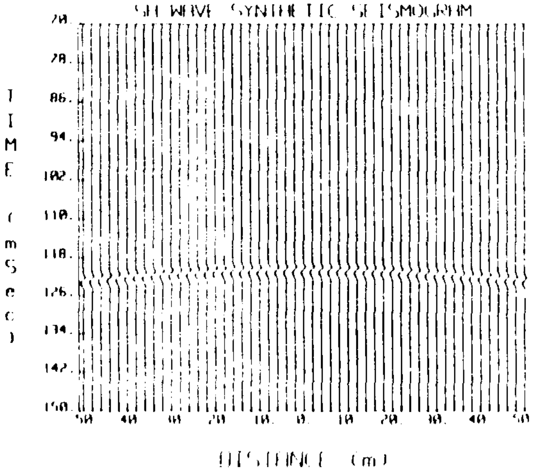
FIGURE G-13. SYNTHETIC SEISMOGRAMS OF SH-WAVES FROM A 2-m DIAMETER CYLINDRICAL CAVITY LOCATED IN A TWO-LAYER LOSSY HALF-SPACE AT A DEPTH OF 100 m. THE CONSTITUTIVE PARAMETERS OF A 5-m THICKNESS SURFACE LAYER AND THE BEDROCK ARE c_1 = 200 m/sec, ρ_1 = 1500 kg/m^3, c_2 = 2500 m/sec, AND ρ_2 = 2700 kg/m^3. THE QUALITY FACTORS OF THE SURFACE LAYER AND THE BEDROCK ARE Q_1 = 50 AND Q_2 = 100, RESPECTIVELY. THE SOURCE IS IN THE BEDROCK, AND THE ARRAY OF DETECTORS HAVING A DETECTOR SPACING OF 2 m IS LOCATED AT THE SURFACE OF THE EARTH.



(a) Total SH-Wave Response

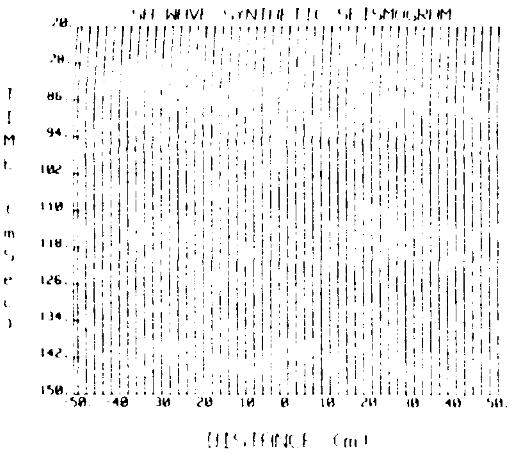


(b) Direct SH-Wave Response

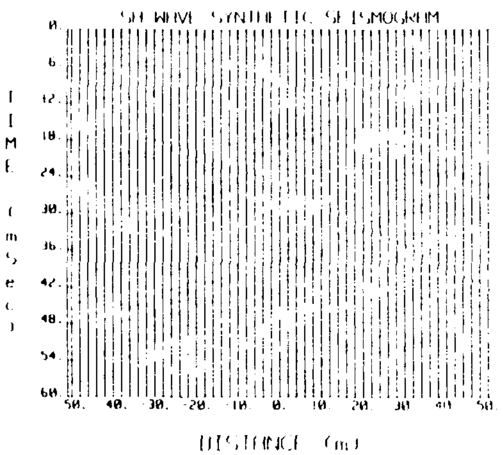


(c) Scattered SH-Wave Response

FIGURE G-14. SYNTHETIC SEISMOGRAMS OF SH-WAVES FROM A 2-m DIAMETER CYLINDRICAL CAVITY LOCATED IN A TWO-LAYER LOSSY HALF-SPACE AT A DEPTH OF 100 m. THE CONSTITUTIVE PARAMETERS OF A 10-m THICKNESS SURFACE LAYER AND THE BEDROCK ARE $c_1 = 200$ m/sec, $\rho_1 = 1500$ kg/m^3, $c_2 = 2500$ m/sec, AND $\rho_2 = 2700$ kg/m^3. THE QUALITY FACTORS OF THE SURFACE LAYER AND THE BEDROCK ARE $Q_1 = 50$ AND $Q_2 = 100$, RESPECTIVELY. THE SOURCE IS IN THE BEDROCK, AND THE ARRAY OF DETECTORS HAVING A DETECTOR SPACING OF 2 m IS LOCATED AT THE SURFACE OF THE EARTH.

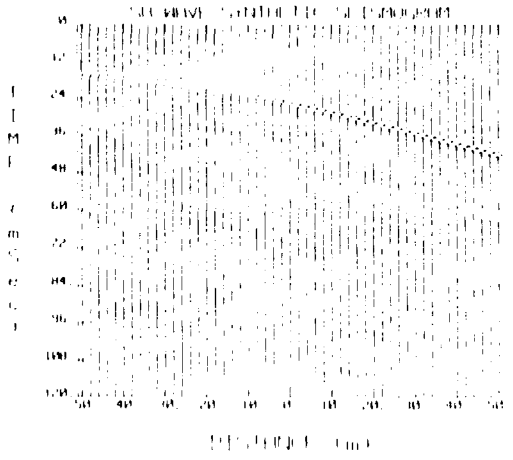


(a) Total SH-Wave Response

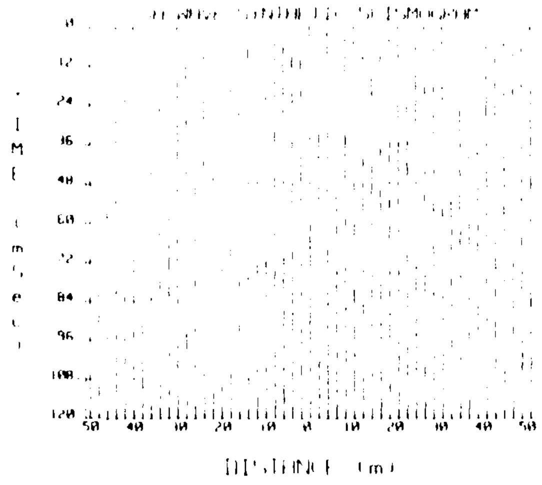


(b) Direct SH-Wave Response

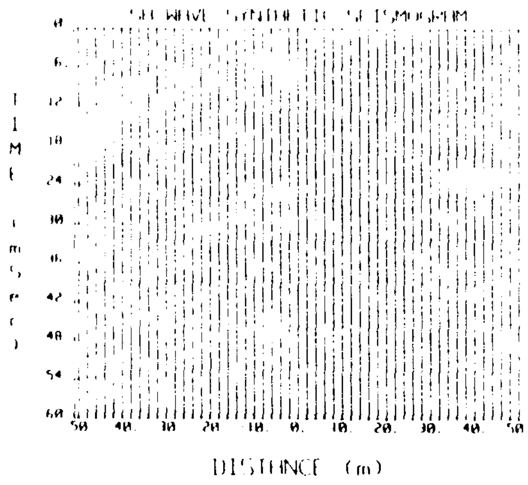
FIGURE G-15. SYNTHETIC SEISMOGRAMS OF SH-WAVES FROM A 2-m DIAMETER CYLINDRICAL CAVITY LOCATED IN A TWO-LAYER LOSSY HALF-SPACE AT A DEPTH OF 100 m. THE CONSTITUTIVE PARAMETERS OF A 5-m THICKNESS SURFACE LAYER AND THE BEDROCK ARE $c_1 = 200$ m/sec, $\rho_1 = 1500$ kg/m^3, $c_2 = 2500$ m/sec, AND $\rho_2 = 2700$ kg/m^3. THE QUALITY FACTORS OF THE SURFACE LAYER AND THE BEDROCK ARE $Q_1 = 10$ AND $Q_2 = 100$, RESPECTIVELY. THE SOURCE IS IN THE BEDROCK, AND THE ARRAY OF DETECTORS HAVING A DETECTOR SPACING OF 2 m IS LOCATED AT THE SURFACE OF THE EARTH.



(c) Scattered SH-Wave Response

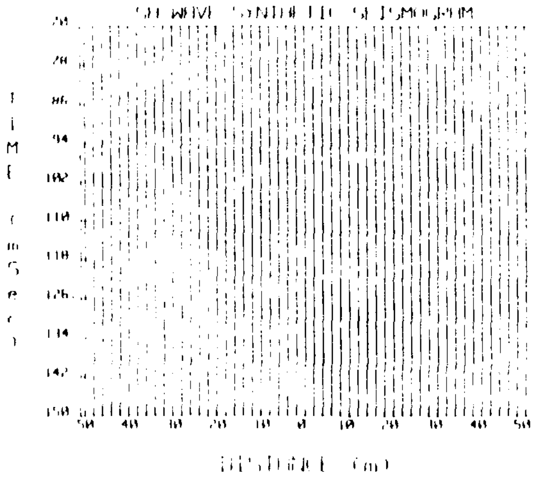


(a) Total SH-Wave Response

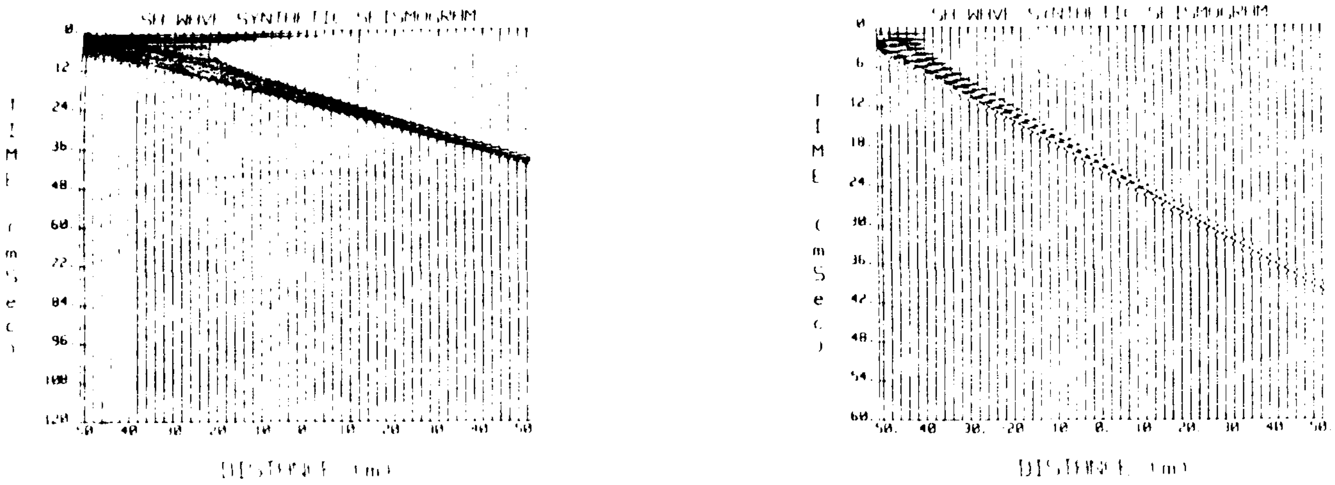


(b) Direct SH-Wave Response

FIGURE G-16. SYNTHETIC SEISMOGRAMS OF SH-WAVES FROM A 2-m DIAMETER CYLINDRICAL CAVITY LOCATED IN A TWO-LAYER LOSSY HALF-SPACE AT A DEPTH OF 100 m. THE CONSTITUTIVE PARAMETERS OF A 10-m THICKNESS SURFACE LAYER AND THE BEDROCK ARE $c_1 = 200$ m/sec, $\rho_1 = 1500$ kg/m^3, $c_2 = 2500$ m/sec, AND $\rho_2 = 2700$ kg/m^3. THE QUALITY FACTORS OF THE SURFACE LAYER AND THE BEDROCK ARE $Q_1 = 10$ AND $Q_2 = 100$, RESPECTIVELY. THE SOURCE IS IN THE BEDROCK, AND THE ARRAY OF DETECTORS HAVING A DETECTOR SPACING OF 2 m IS LOCATED AT THE SURFACE OF THE EARTH.

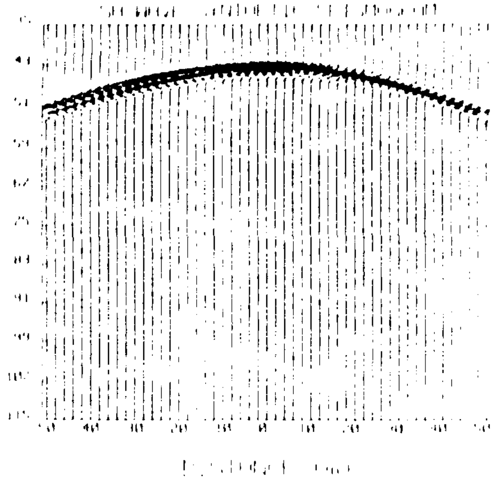


(c) Scattered SH-Wave Response



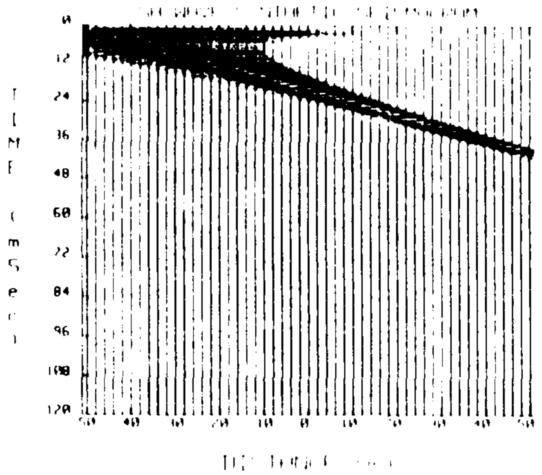
(a) Total SH-Wave Response

(b) Direct SH-Wave Response

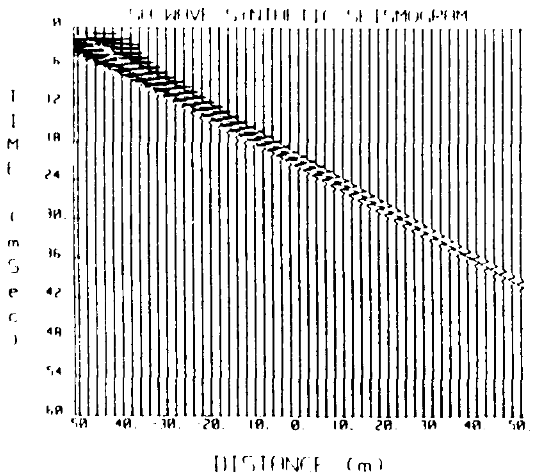


(c) Scattered SH-Wave Response

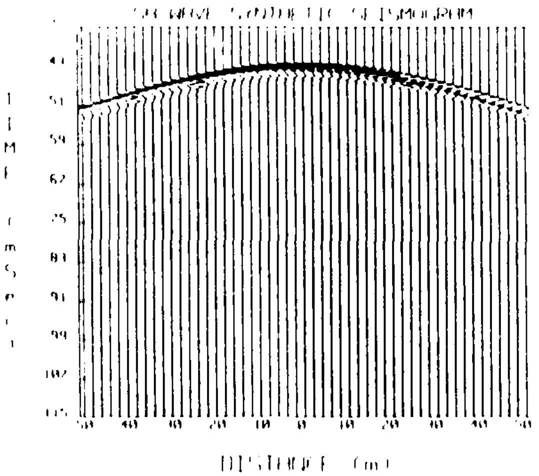
FIGURE G-17. SYNTHETIC SEISMOGRAMS OF SH-WAVES FROM A 2-m DIAMETER CYLINDRICAL CAVITY LOCATED IN A TWO-LAYEP LOSSY HALF-SPACE AT A DEPTH OF 50 m. THE SEISMOGRAMS SHOW ONLY THE DIRECT SH-WAVE PARTICLE MOTION AND THE FIRST TUNNEL TARGET REFLECTION. THE CONSTITUTIVE PARAMETERS OF A 5-m THICKNESS SURFACE LAYER AND THE BEDROCK ARE c_1 = 500 m/sec, ρ_1 = 1500 m/sec^3, c_2 = 2500 m/sec, AND ρ_2 = 2700 kg/m^3. THE QUALITY FACTORS OF THE SURFACE LAYER AND THE BEDROCK ARE Q_1 = 50 AND Q_2 = 100, RESPECTIVELY. THE SOURCE AND THE ARRAY OF DETECTORS HAVING A DETECTOR SPACING OF 2 m ARE LOCATED IN THE BEDROCK.



(a) Total SH-Wave Response

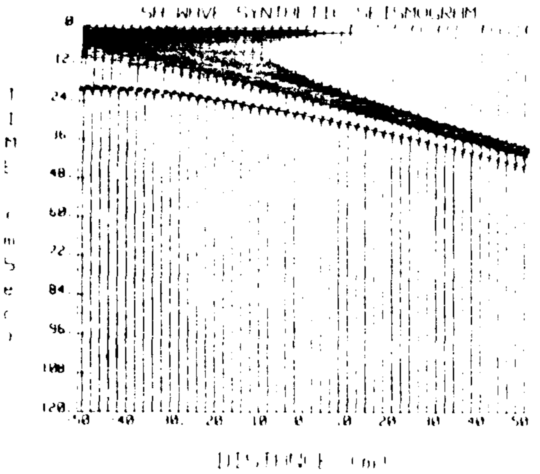


(b) Direct SH-Wave Response

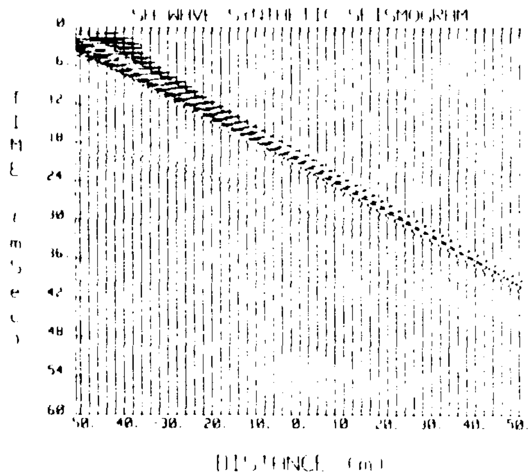


(c) Scattered SH-Wave Response

FIGURE G-18. SYNTHETIC SEISMOGRAMS OF SH-WAVES FROM A 2-m DIAMETER CYLINDRICAL CAVITY LOCATED IN A TWO-LAYER LOSSY HALF-SPACE AT A DEPTH OF 50 m. THE SEISMOGRAMS SHOW THE DIRECT SH-WAVE PARTICLE MOTION PLUS THE FIRST LAYER REFLECTION AND THE FIRST AND SECOND TUNNEL TARGET REFLECTIONS. THE CONSTITUTIVE PARAMETERS OF A 5-m THICKNESS SURFACE LAYER AND THE BEDROCK ARE c_1 = 500 m/sec, ρ_1 = 1500 m/sec, c_2 = 2500 m/sec, AND ρ_2 = 2700 kg/m^3. THE QUALITY FACTORS OF THE SURFACE LAYER AND THE BEDROCK ARE Q_1 = 50 AND Q_2 = 100, RESPECTIVELY. THE SOURCE AND THE ARRAY OF DETECTORS HAVING A DETECTOR SPACING OF 2 m ARE LOCATED IN THE BEDROCK.

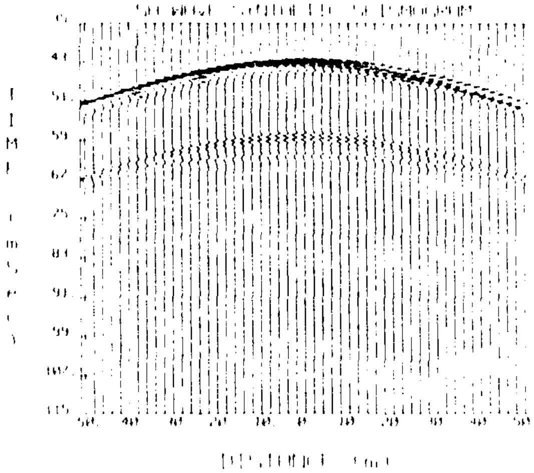


(a) Total SH-Wave Response

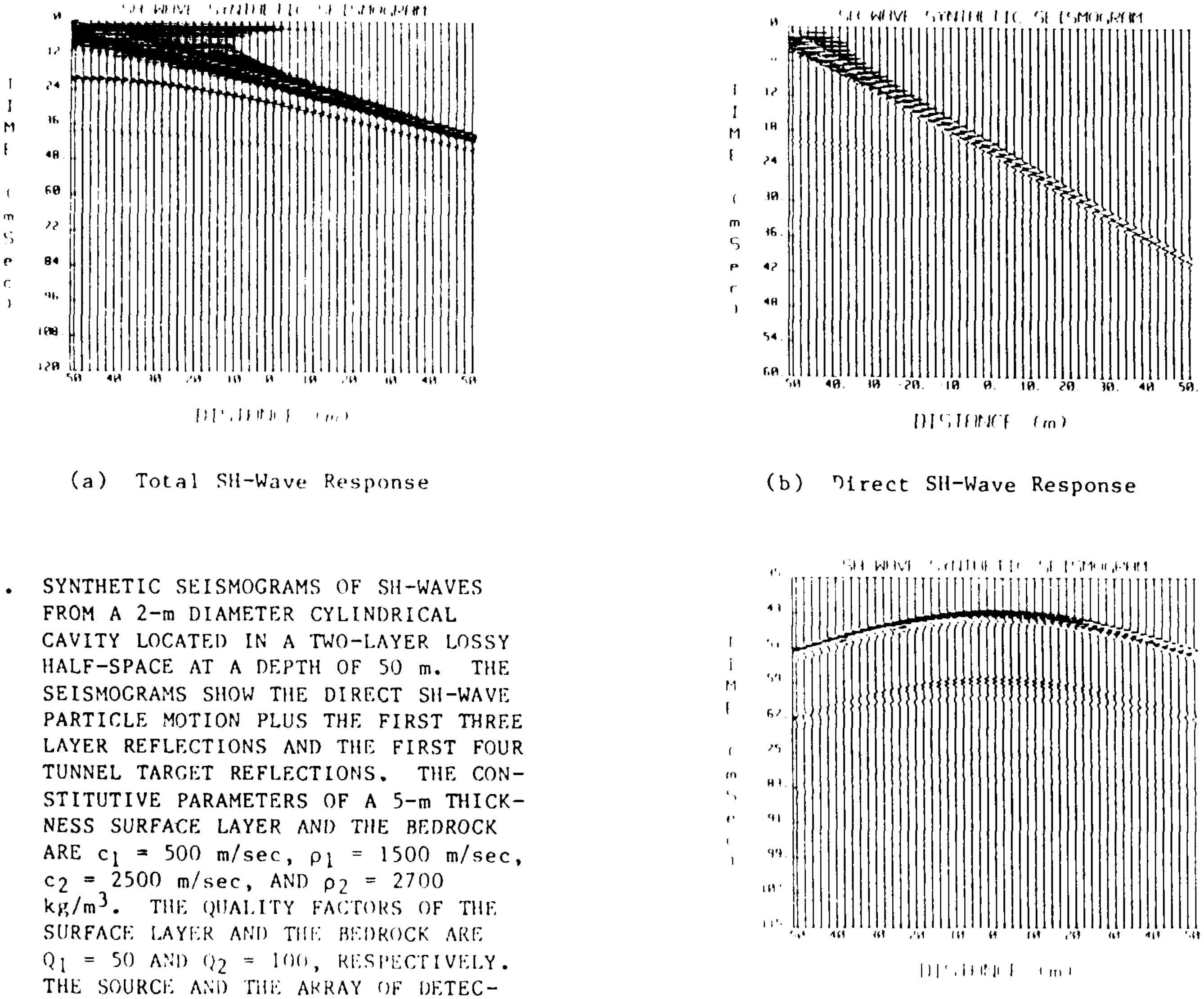


(b) Direct SH-Wave Response

FIGURE G-19. SYNTHETIC SEISMOGRAMS OF SH-WAVES *FROM A 2-m DIAMETER CYLINDRICAL* CAVITY LOCATED IN A TWO-LAYER LOSSY HALF-SPACE AT A DEPTH OF 50 m. THE SEISMOGRAMS SHOW THE DIRECT SH-WAVE PARTICLE MOTION PLUS THE FIRST AND SECOND LAYER REFLECTIONS AND THE FIRST THREE TUNNEL TARGET REFLECTIONS. THE CONSTITUTIVE PARAMETERS OF A 5-m THICKNESS SURFACE LAYER AND THE BEDROCK ARE c_1 = 500 m/sec, ρ_1 = 1500 m/sec, c_2 = 2500 m/sec, AND ρ_2 = 2700 kg/m^3. THE QUALITY FACTORS OF THE SURFACE LAYER AND THE BEDROCK ARE Q_1 = 50 AND Q_2 = 100, RESPECTIVELY. THE SOURCE AND THE ARRAY OF DETECTORS HAVING A DETECTOR SPACING OF 2 m ARE LOCATED IN THE BEDROCK.



(c) Scattered SH-Wave Response

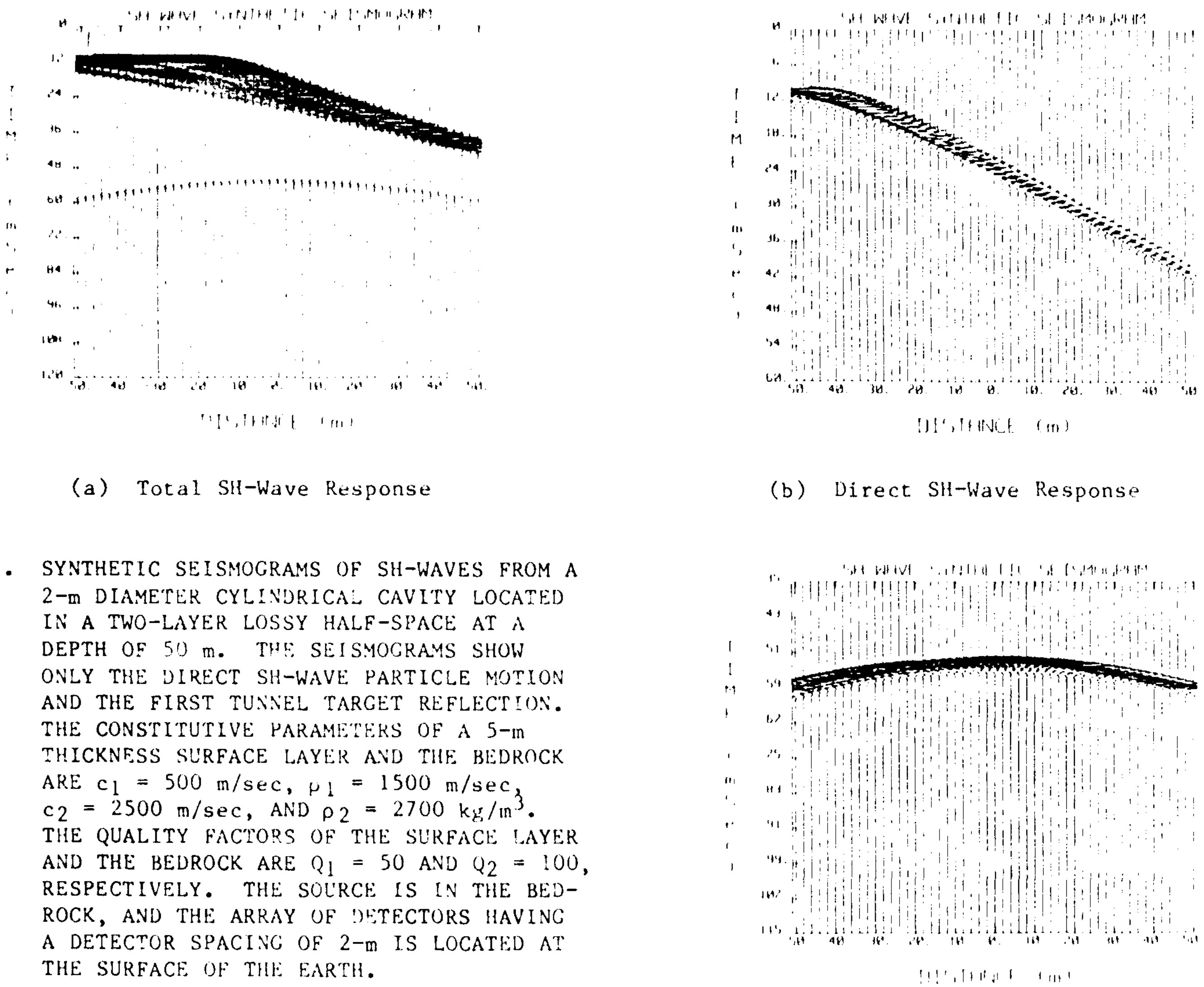


(a) Total SH-Wave Response

(b) Direct SH-Wave Response

(c) Scattered SH-Wave Response

FIGURE G-20. SYNTHETIC SEISMOGRAMS OF SH-WAVES FROM A 2-m DIAMETER CYLINDRICAL CAVITY LOCATED IN A TWO-LAYER LOSSY HALF-SPACE AT A DEPTH OF 50 m. THE SEISMOGRAMS SHOW THE DIRECT SH-WAVE PARTICLE MOTION PLUS THE FIRST THREE LAYER REFLECTIONS AND THE FIRST FOUR TUNNEL TARGET REFLECTIONS. THE CONSTITUTIVE PARAMETERS OF A 5-m THICKNESS SURFACE LAYER AND THE BEDROCK ARE c_1 = 500 m/sec, ρ_1 = 1500 m/sec, c_2 = 2500 m/sec, AND ρ_2 = 2700 kg/m^3. THE QUALITY FACTORS OF THE SURFACE LAYER AND THE BEDROCK ARE Q_1 = 50 AND Q_2 = 100, RESPECTIVELY. THE SOURCE AND THE ARRAY OF DETECTORS HAVING A DETECTOR SPACING OF 2 m ARE LOCATED IN THE BEDROCK.

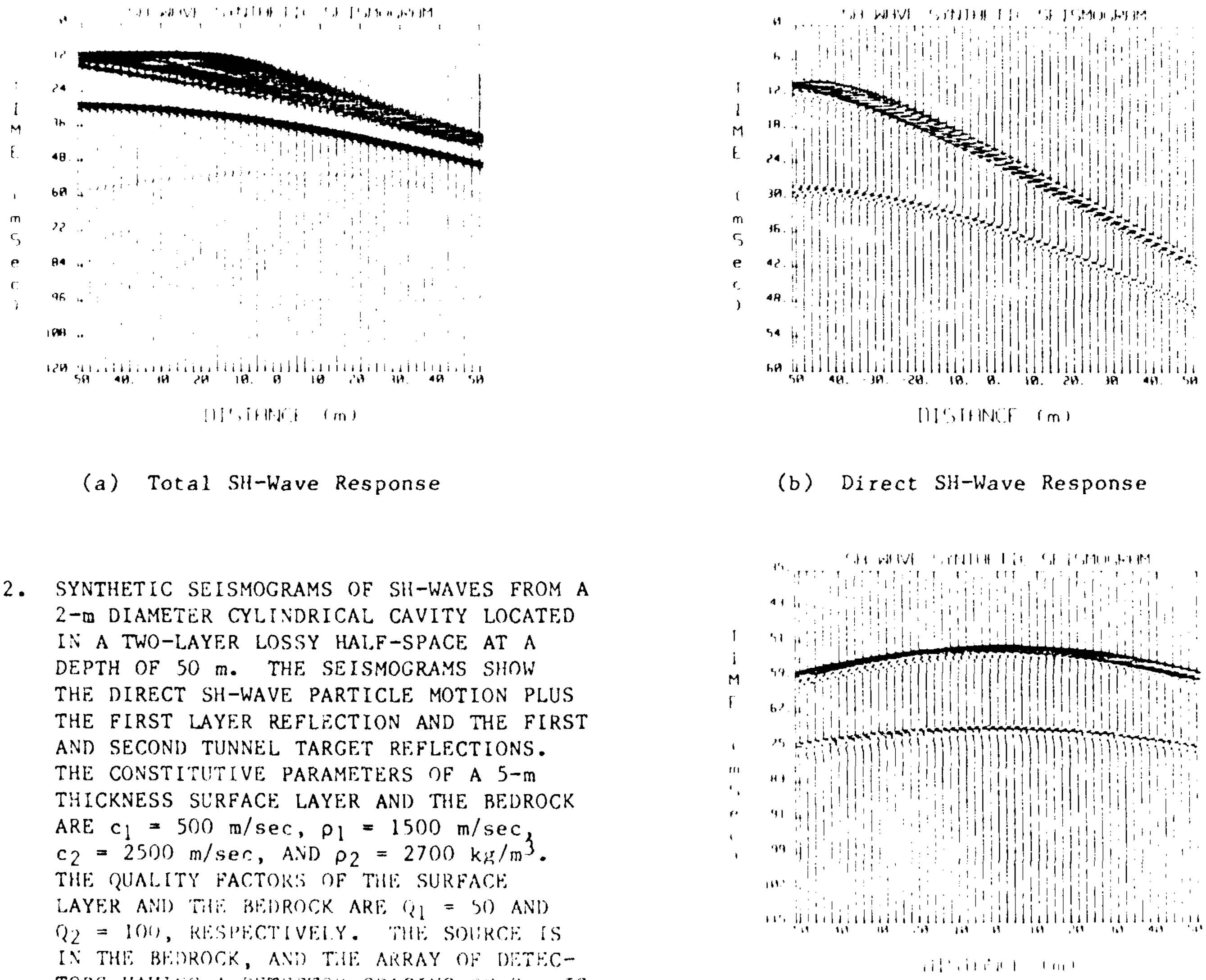


(a) Total SH-Wave Response

(b) Direct SH-Wave Response

(c) Scattered SH-Wave Response

FIGURE G-21. SYNTHETIC SEISMOGRAMS OF SH-WAVES FROM A 2-m DIAMETER CYLINDRICAL CAVITY LOCATED IN A TWO-LAYER LOSSY HALF-SPACE AT A DEPTH OF 50 m. THE SEISMOGRAMS SHOW ONLY THE DIRECT SH-WAVE PARTICLE MOTION AND THE FIRST TUNNEL TARGET REFLECTION. THE CONSTITUTIVE PARAMETERS OF A 5-m THICKNESS SURFACE LAYER AND THE BEDROCK ARE c_1 = 500 m/sec, ρ_1 = 1500 m/sec, c_2 = 2500 m/sec, AND ρ_2 = 2700 kg/m^3. THE QUALITY FACTORS OF THE SURFACE LAYER AND THE BEDROCK ARE Q_1 = 50 AND Q_2 = 100, RESPECTIVELY. THE SOURCE IS IN THE BEDROCK, AND THE ARRAY OF DETECTORS HAVING A DETECTOR SPACING OF 2-m IS LOCATED AT THE SURFACE OF THE EARTH.

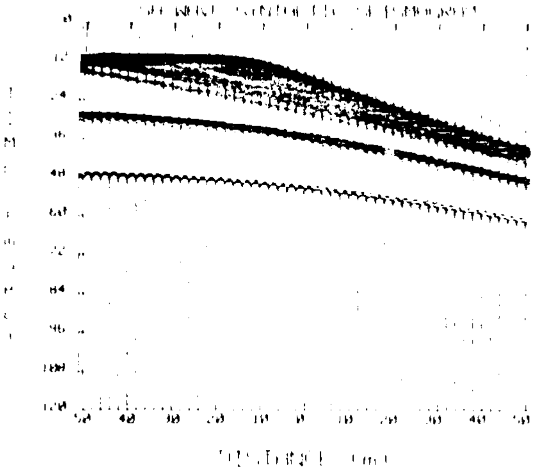


(a) Total SH-Wave Response

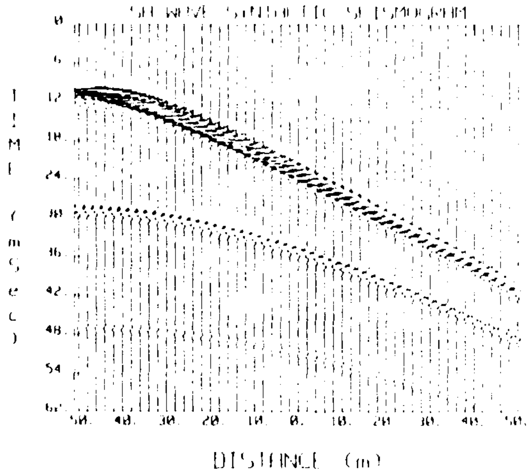
(b) Direct SH-Wave Response

(c) *Scattered SH-Wave Response*

FIGURE G-22. SYNTHETIC SEISMOGRAMS OF SH-WAVES FROM A 2-m DIAMETER CYLINDRICAL CAVITY LOCATED IN A TWO-LAYER LOSSY HALF-SPACE AT A DEPTH OF 50 m. THE SEISMOGRAMS SHOW THE DIRECT SH-WAVE PARTICLE MOTION PLUS THE FIRST LAYER REFLECTION AND THE FIRST AND SECOND TUNNEL TARGET REFLECTIONS. THE CONSTITUTIVE PARAMETERS OF A 5-m THICKNESS SURFACE LAYER AND THE BEDROCK ARE c_1 = 500 m/sec, ρ_1 = 1500 m/sec, c_2 = 2500 m/sec, AND ρ_2 = 2700 kg/m^3. THE QUALITY FACTORS OF THE SURFACE LAYER AND THE BEDROCK ARE Q_1 = 50 AND Q_2 = 100, RESPECTIVELY. THE SOURCE IS IN THE BEDROCK, AND THE ARRAY OF DETECTORS HAVING A DETECTOR SPACING OF 2-m IS LOCATED AT THE SURFACE OF THE EARTH.

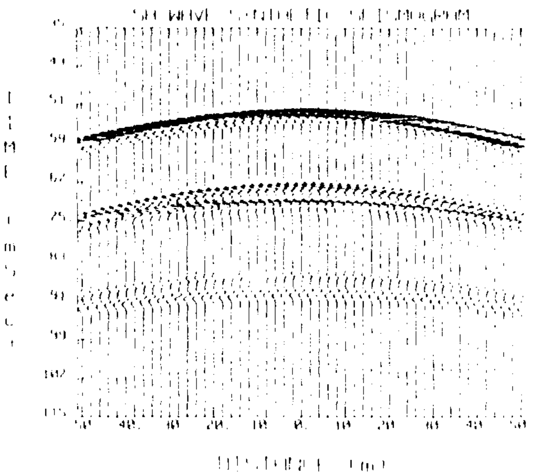


(a) Total SH-Wave Response

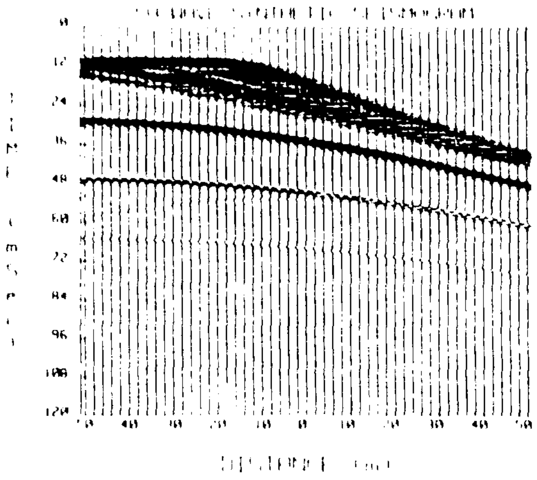


(b) Direct SH-Wave Response

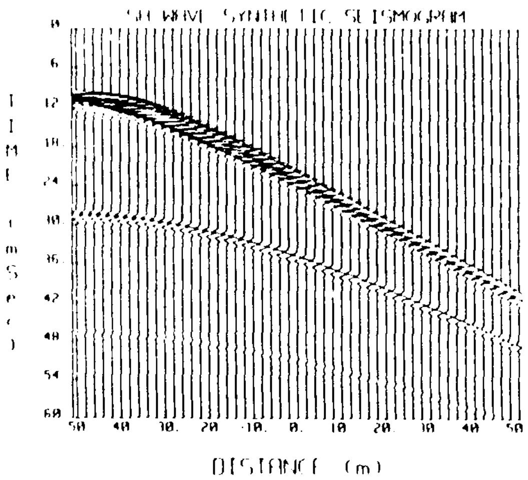
FIGURE G-23. SYNTHETIC SEISMOGRAMS OF SH-WAVES FROM A 2-m DIAMETER CYLINDRICAL CAVITY LOCATED IN A TWO-LAYER LOSSY HALF-SPACE AT A DEPTH OF 50 m. THE SEISMOGRAMS SHOW THE DIRECT SH-WAVE PARTICLE MOTION PLUS THE FIRST AND SECOND LAYER REFLECTIONS AND THE FIRST THREE TUNNEL TARGET REFLECTIONS. THE CONSTITUTIVE PARAMETERS OF A 5-m THICKNESS SURFACE LAYER AND THE BEDROCK ARE c_1 = 500 m/sec, ρ_1 = 1500 m/sec, c_2 = 2500 m/sec, AND ρ_2 = 2700 kg/m^3. THE QUALITY FACTORS OF THE SURFACE LAYER AND THE BEDROCK ARE Q_1 = 50 AND Q_2 = 100, RESPECTIVELY. THE SOURCE IS IN THE BEDROCK, AND THE ARRAY OF DETECTORS HAVING A DETECTOR SPACING OF 2-m IS LOCATED AT THE SURFACE OF THE EARTH.



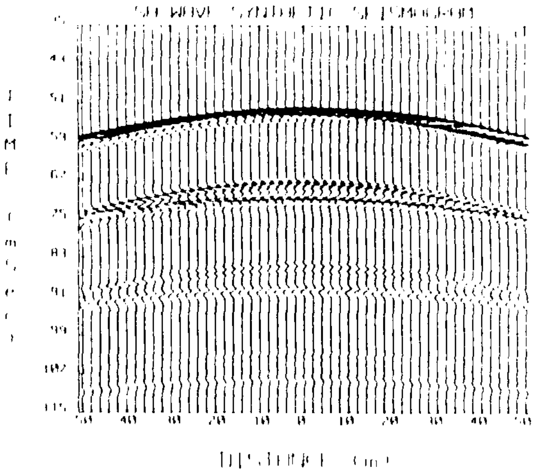
(c) Scattered SH-Wave Response



(a) Total SH-Wave Response



(b) Direct SH-Wave Response



(c) Scattered SH-Wave Response

FIGURE G-24. SYNTHETIC SEISMOGRAMS OF SH-WAVES FROM A 2-m DIAMETER CYLINDRICAL PARTICLE CAVITY LOCATED IN A TWO-LAYER LOSSY HALF-SPACE AT A DEPTH OF 50 m. THE SEISMOGRAMS SHOW THE DIRECT SH-WAVE MOTION PLUS THE FIRST THREE LAYER REFLECTIONS AND THE FIRST FOUR TUNNEL TARGET REFLECTIONS. THE CONSTITUTIVE PARAMETERS OF A 5-m THICKNESS SURFACE LAYER AND THE BEDROCK ARE c_1 = 500 m/sec, ρ_1 = 1500 m/sec, c_2 = 2500 m/sec, AND ρ_2 = 2700 kg/m^3. THE QUALITY FACTORS OF THE SURFACE LAYER AND THE BEDROCK ARE Q_1 = 50 AND Q_2 = 100, RESPECTIVELY. THE SOURCE IS IN THE BEDROCK, AND THE ARRAY OF DETECTORS HAVING A DETECTOR SPACING OF 2-m IS LOCATED AT THE SURFACE OF THE EARTH.

APPENDIX H

COMPUTER PROGRAMS

APPENDIX H

Computer Programs

H.1 Plane Wave Scattering

The programs in this Section H.1 model the scattering of plane SH waves from a cylindrical cavity in a homogeneous lossless whole-space as a function of frequency. All computational programs are written in FORTRAN 77 language. The graphics programs of Section H.3 are written in C language.

H.1.1 Energy Scattering Cross-Section Program

```
C
C   Program     : csect
C
C   Author      : J.C. Biard
C
C   Discussion : The program csect calculates the total energy-scattering
C      cross-section as expressed by Equation (37) in Volume I as a function
C      of frequency for a cylindrical cavity of radius, a, illuminated by a
C      plane SH wave in a homogeneous whole space.  The cross-section is
C      reported as a function of wavenumber multiplied by the tunnel radius.
C      The summation of the contributions made by the different scattering
C      modes is computed to an accuracy of 0.01 percent of the value.
C
C   Invocation : The program is invoked by typing
C
C            csect > ofile
C
C      where ofile is the output filename.
C
C   Formal Arguments : none
C
C   Procedures called from this module :
C
C      jn                  Bessel function of the first kind. (HP-UX utility.)
C      yn                  Bessel function of the second kind. (HP-UX utility.)
C
C   External Variables : none

$ALI
$ALIAS jn = 'jn' (%val, %val)
    AS yn = 'yn' (%val, %val)

      program csect

      implicit none

      integer*4 stdin, stdout, stderr
      integer*4   j, k, l, m, n
      real*8      sect, ka, x, acc
      complex*16 aj, i

      real*8 jn, yn, re, im

      data stdin, stdout, stderr / 5, 6, 6/

C
C  Define statement functions for the real and imaginary parts of the mode
C     contribution terms.
C
      re(n, x) = n * jn(n, x) - x * jn((n+1), x)
      im(n, x) = n * yn(n, x) - x * yn((n+1), x)

      i = cmplx(0., 1.)

C
C  Explicitly write the zero frequency cross-section value.
C
      sect = 0.
      ka = 0.
      n = 0
      acc = 0.
      write(stdout, 2000) ka, sect, n, acc

C
C  For ka = .05 to 2. by .05, calculate the cross-section.
C
      do k = 1, 40
         j = 0
         ka = k * .05
C
C  Get the mode contribution term for j = 0.
C
         aj = -dcmplx(re(j, ka)) / dcmplx(re(j, ka), im(j, ka))
         sect = real(aj * conjg(aj))
C
C  Loop over j to sum the contributions.
C
         do j = 1, 50
C
C  Get the jth contribution to the cross-section.
C
            aj = dcmplx(re(j, ka)) / dcmplx(re(j, ka), im(j, ka))
            x = 2. * real(aj * conjg(aj))
C
C  Calculate the accuracy test value and add the new term to the
C     sum for the cross-section.
C
            acc = abs(x / sect)
            sect = sect + x
            n = j
```

```
C
C   If the sum has at least 15 terms and the accuracy is in limits, then
C      stop the sum.
C
            if (j .ge. 15 .and. acc .le. .0001) go to 10
         end do
   10    continue
C
C   Multiply the cross-section sum by constant terms.
C
         sect = sect * 2. / ka
C
C   If the accuracy is out of limits then write an error message.
C
         if (acc .gt. .0001) then
            write(stderr, 2001)
         end if
C
C   Write the results to the standard output.
C
         write(stdout, 2000) ka, sect, n, acc
      end do
C
C   For ka = 2.1 to 10. by .1, calculate the cross-section.
C
      do k = 0, 80
         j = 0
         ka = 2. + k * .1
C
C   Get the mode contribution term for j = 0.
C
         aj = -dcmplx(re(j, ka)) / dcmplx(re(j, ka), im(j, ka))
         sect = real(aj * conjg(aj))
C
C   Loop over j to sum the contributions.
C
         do j = 1, 50
C
C   Get the jth contribution to the cross-section.
C
            aj = dcmplx(re(j, ka)) / dcmplx(re(j, ka), im(j, ka))
            x = 2. * real(aj * conjg(aj))
C
C   Calculate the accuracy test value and add the new term to the
C      sum for the cross-section.
C
            acc = abs(x / sect)
            sect = sect + x
            n = j
C
C   If the sum has at least 15 terms and the accuracy is in limits, then
C      stop the sum.
C
```

```
            if (j .ge. 15 .and. acc .le. .0001) go to 20
         end do
   20    continue
C
C   Multiply the cross-section sum by constant terms.
C
         sect = sect * 2. / ka
C
C   If the accuracy is out of limits then write an error message.
C
         if (acc .gt. .0001) then
            write(stderr, 2001)
         end if
C
C   Write the results to the standard output.
C
         write(stdout, 2000) ka, sect, n, acc
      end do

      stop

 2000 format("ka, sect, n, acc = ",
     1       g20.10, ", ", g20.10,", ", i20,", ", g20.10)
 2001 format("+++++++++++ Not in accuracy limits +++++++++++")

      end
```

H.1.2 Displacement Amplitude Programs

H.1.2.1 dispa

```
C
C   Program    : dispa
C
C   Author     : J.C. Biard
C
C   Discussion : This program calculates the total displacement field as
C      expressed by Equation (26) in Volume I at the surface of a cylindrical
C      cavity of radius a for an incident plane SH-wave of unit amplitude with
C      wave number k.
C      The results are given as the relative amplitude and phase at 45, 90, and
C      135 degrees for values of k between 0. and 10.
C
C
C   Invocation : The program is invoked by typing
C
C           dispa > ofile
C
C      where ofile is the output filename.
C
C   Formal Arguments : none
```

```
C   Procedures called from this module :
C
C      plane_scatter          Subroutine that calculates the displacement
C                                amplitude.
C
C   External Variables : none
C

      program dispa

      implicit none

      integer*4  stdin, stdout, stderr

      integer*4  k, n
      real*8     amp, phase, phi, phi0, phiend, ka, kr, acc, pi
      real*8     oldphase, offset, phasemax, accl, krc
      complex*16 dsp
      character*1 ff

      data stdin, stdout, stderr / 5, 6, 6/
      data pi / 3.14159265358979/
      data acc / .0001/

      ff = char(12)
C
C   Set the angle limits and step.
C
      phi0 = 45. * pi / 180.
      phiend = 3. * phi0
      phasemax = 7. * pi / 4.

C
C   Loop over the angles of interest.
C
      do phi = phi0, phiend, phi0
         write(stdout, 2000) ff, phi
         offset = 0.
         oldphase = 0.
C
C      Calculate values for 0. <= ka <= 2., stepping by .05.
C
         do ka = 0., 2.01, .05
            kr = ka
C
C         Get scattered field value.
C
            call plane_scatter(acc, ka, kr, phi, n, accl, dsp)
C
C         Add incident field term.
C
            krc = kr * cos(phi)
            dsp = dsp + dcmplx(cos(krc), sin(krc))
```

```
C
C         Find amplitude and phase.
C
            amp = abs(dsp)
            phase = datan2(dimag(dsp), dble(dsp))
C
C         If phase has wrapped around the 180  or 360 deg. point, unwrap it.
C
            if (abs(oldphase - phase) .gt. phasemax) then
               offset = offset - sign(2.*pi, phase)
            end if
            oldphase = phase
            phase = phase + offset - krc
C
C         Write out results.
C
            write(stdout, 2001) n, ka, amp, phase, accl
         end do
C
C      Calculate values for 2.1 <= ka <= 10., stepping by .1.
C
         do ka = 2.1, 10.01, .1
            kr = ka
C
C         Get scattered field value.
C
            call plane_scatter(acc, ka, kr, phi, n, accl, dsp)
C
C         Add incident field term.
C
            krc = kr * cos(phi)
            dsp = dsp + dcmplx(cos(krc), sin(krc))
C
C         Find amplitude and phase.
C
            amp = abs(dsp)
            phase = datan2(dimag(dsp), dble(dsp))
C
C         If phase has wrapped around the 180  or 360 deg. point, unwrap it.
C
            if (abs(oldphase - phase) .gt. phasemax) then
               offset = offset - sign(2.*pi, phase)
            end if
            oldphase = phase
            phase = phase + offset - krc
C
C         Write out results.
C
            write(stdout, 2001) n, ka, amp, phase, accl
         end do
      end do

      stop
```

```
 2000 format(a1, "SH Amplitude and Phase at r = a, ", "phi = ", g20.10
     1    // t2, "n", t15, "ka", t35, "amp", t54, "phase", t76, "acc"/)
 2001 format(i4, " ", g20.10, " ", g20.10," ", g20.10," ", g20.10)

      end
```

H.1.2.2 dispb

```
C
C   Program    : dispb
C
C   Author     : J.C. Biard
C
C   Discussion : This program calculates the total displacement field for the
C      scattering of a plane SH-wave from a cylindrical cavity of radius, a,
C      at a wave number of ka.
C      The field is determined at a variety of distances from the cylinder
C      and at angles to the incident wave propagation vector of 45, 90, and 135
C      degrees.  The wave number also takes on values of .5, 1., and 2.
C
C   Invocation : The program is invoked by typing
C
C            dispb > ofile
C
C      where ofile is the output file name.
C
C   Formal Arguments : none
C
C   Procedures called from this module :
C
C      plane_scatter              Subroutine that calculates the displacement
C                                    amplitude.
C
C   External Variables : none
C

      program dispb

      implicit none

      integer*4  stdin, stdout, stderr

      integer*4  k, n, nka
      real*8     amp, phi, phi0, phiend, ka, kr, acc, pi
      real*8     roa, accl, krc, kas(3)
      complex*16 dsp
      character*1 ff
      data stdin, stdout, stderr / 5, 6, 6/
      data pi / 3.1415926535897/
      data nka, kas / 3, .5, 1., 2./
      data acc / .0001/

      ff = char(12)
C
C    Set the angle limits and step.
C
      phi0 = 45. * pi / 180.
      phiend = 3. * phi0

C
C    Loop over the angles.
C
      do phi = phi0, phiend, phi0
C
C       Loop over the ka values.
C
         do k = 1, nka
            ka = kas(k)
            write(stdout, 2000) ka, phi
C
C          Loop over 1. <= r/a <= 10., stepping by .5.
C
            do roa = 1., 10., .5
               kr = roa * ka
C
C       Get scattered field value.
C
               call plane_scatter(acc, ka, kr, phi, n, accl, dsp)
C
C       Add incident field term.
C
               krc = kr * cos(phi)
               dsp = dsp + dcmplx(cos(krc), sin(krc))
C
C       Find amplitude.
C
               amp = abs(dsp)
C
C       Write out results.
C
               write(stdout, 2001) n, roa, kr, amp, accl
            end do
C
C          Loop over 11. <= r/a <= 50., stepping by 1.
C
            do roa = 11., 50., 1.
               kr = roa * ka
C
C       Get scattered field value.
C
               call plane_scatter(acc, ka, kr, phi, n, accl, dsp)
```

```
C
C      Add incident field term.
C
               krc = kr * cos(phi)
               dsp = dsp + dcmplx(cos(krc), sin(krc))
C
C      Find amplitude.
C
               amp = abs(dsp)
C
C      Write out results.
C
               write(stdout, 2001) n, roa, kr, amp, accl
            end do
         end do
      end do

      stop

 2000 format(a1, "Displacement Amplitude, ka = ", g20.10, "phi = ", g20.10
     1    // t2, "n", t15, "r/a", t35, "kr", t54, "amp", t76, "acc"/)
 2001 format(i4, " ", g20.10, " ", g20.10," ", g20.10," ", g20.10)

      end
```

H.1.2.3 disped

```
C
C   Program     : disped
C
C   Author      : J.C. Biard
C
C   Discussion : This program calculates the amplitude of the displacement
C      field as expressed by Equation (26) in Volume I as a function of angle
C      for the scattering of a plane SH-wave from a cylindrical cavity of
C      radius a.
C      The values are determined for a number of cases of radial
C      distance from the cylinder center and for a number of values of the
C      wave number.
C         An option is provided in the code where it is possible to comment
C      out a section and obtain the scattered field only.
C
C   Invocation : The program is invoked by typing
C
C            disped > ofile
C
C      where ofile is the output filename.
C
C   Formal Arguments : none
```

```
C   Procedures called from this module :
C
C      plane_scatter                     Subroutine that calculates the
C                                           displacement amplitude.
C
C   External Variables : none
C

       program dispcd

       implicit none

       integer*4  stdin, stdout, stderr

       integer*4  k, m, n, nka, nra
       real*8     amp, phi, phi0, ka, kr, acc, pi, accl
       real*8     roa, ra(10), krc, kas(10)
       complex*16 dsp
       character*1 ff

       data stdin, stdout, stderr / 5, 6, 6/
       data pi / 3.1415926535897/
       data nka, kas / 3, .5, 1., 2./
       data nra, ra / 7, 1., 2., 5., 10., 20., 50., 100./
       data acc / .0001/

       ff = char(12)
       phi0 = pi / 180.

C
C   Loop over the r/a.
C
       do k = 1, nra
          roa = ra(k)
C
C      Find the ka and kr values.
C
          do m = 1, nka
             ka = kas(m).
             kr = roa * ka
             write(stdout, 2000) ff, ka, roa
C
C         Loop over the angles.
C
             do phi = 0., pi, phi0
C
C      Get scattered field value.
C
                call plane_scatter(acc, ka, kr, phi, n, accl, dsp)
C
C      Add incident field term.
C         If the following two lines are commented, the scattered field only
C         will be described.
C
```

```
                krc = kr * cos(phi)
                dsp = dsp + dcmplx(cos(krc), sin(krc))
C
C      Find amplitude
C
                amp = abs(dsp)
C
C      Write out results.
C
                write(stdout, 2001) n, phi, amp, accl
             end do
          end do
       end do

       stop

 2000 format(a1,"Displacement, ka = ", g20.10, "r/a = ", g20.10
     1    // t2, "n", t15, "phi", t35, "amp", t54, "acc"/)
 2001 format(i4, " ", g20.10, " ", g20.10," ", g20.10)

       end
```

H.1.2.4 dispe

```
C
C   Program    : dispe
C
C   Author     : J.C. Biard
C
C   Discussion : This program calculates the amplitude of the displacement
C     field as expressed by Equation (26) in Volume I from a plane SH-wave of
C     unit amplitude incident on a cylindrical cavity of radius, a, and at a
C     wave number of k.
C     The values are determined along the x-axis at intervals of dxl for a
C     cylinder located at x=0. and y = -h, and for an incident wave whose
C     propagation vector makes an angle -theta with the negative y-axis.
C        The propagation vector makes an angle of theta with the y-axis.
C     In this example, the propagation vector is parallel to a line from
C     (-dx, 0.) to (0., -h).
C     The point (-dx, 0.) is the shot point for seismic exploration work.
C     The program also contains an option to set the angle theta
C     independently of the position dx.
C
C   Invocation : The program is invoked by typing
C
C           dispe > ofile
C
C      where ofile is the output filename.
C
C   Formal Arguments : none
C
```

```
C   Procedures called from this module :
C
C      plane_scatter              Subroutine that calculates the
C                                    displacement amplitude.
C
C   External Variables : none
C

      program dispe

      implicit none

      integer*4  stdin, stdout, stderr

      integer*4  k, m, n, nh, nka
      real*8     amp, phase, phi, ka, kr, acc, pi, hoa(10), accl, krc
      real*8     oldphase, offset, phasemax, xl, h, a, dxl, theta, dx
      real*8     kas(10)
      complex*16 dsp
      character*1 ff

      data stdin, stdout, stderr / 5, 6, 6/
      data pi / 3.1415926535897/
      data nka, kas / 3, .5, 1., 2., 7*0./
C     data nh, hoa / 5, 2., 5., 10., 20., 50., 5*0./
      data nh, hoa / 5, 20., 40., 60., 80., 100., 5*0./
C     data nh, hoa / 5, 5., 10., 15., 20., 25., 5*0./
      data a, dxl, dx / 1., .5, 54./
      data acc / .0001/

      ff = char(12)

C
C   Set the value of theta.  (Not in use now.)
C
C     theta = pi / 2.  (For parallel to ground case.)
C     theta = pi       (For hole-to-hole case.)
C
C   Loop over the depths.
C
      do k = 1, nh
         h = hoa(k) * a
C
C      Find the "source point" angle theta.
C         Comment out this line when using a preset theta value.
C
         theta = datan2(dx, h)
C
C      Loop over the values of ka.
C
         do m = 1, nka
            ka = kas(m)
            write(stdout, 2000) ff, ka, h
```

```
C
C         Loop over the points on the detector line.
C
            do xl = -50., 50., dxl
C
C           Calculate the kr and phi values.
C
               kr = ka * sqrt(h**2 + xl**2) / a
               phi = datan2(h, xl) + pi / 2. - theta
C
C      Get scattered field value.
C
               call plane_scatter(acc, ka, kr, phi, n, accl, dsp)
C
C      Add incident field term.
C
               krc = kr * cos(phi)
               dsp = dsp + dcmplx(cos(krc), sin(krc))
C
C      Find amplitude and phase.
C
               amp = abs(dsp)
C              phase = datan2(dimag(dsp), dble(dsp))
C
C      If phase has wrapped around the 180  or 360 deg. point, unwrap it.
C
C              if (abs(oldphase - phase) .gt. phasemax) then
C                 offset = offset - sign(2.*pi, phase)
C              end if
C              oldphase = phase
C              phase = phase + offset - ka * cos(phi)
C
C      Write out results.
C
               write(stdout, 2001) n, xl, amp, accl
            end do
         end do
      end do

      stop

 2000 format(al, "Displacement, ka = ", g20.10, "h = ", g20.10
     1    // t4, "n", t15, "xl", t35, "amp", t54, "acc"/)
 2001 format(i4, " ", g20.10, " ", g20.10," ", g20.10)

      end
```

H.1.2.5 plane_scatter Subroutine

```
C
C   Procedure name : plane_scatter
C
C   Author         : J.C. Biard
C
C   Discussion : This subroutine calculates the scattered displacement
C      field as expressed by Equation (25) in Volume I for a unit amplitude
C      plane SH-wave incident on a cylindrical cavity buried in a
C      whole-space.
C
C   The input arguments are:
C
C      acc       The fractional accuracy level used in the convergence test.
C       a        The wave number multiplied by the cylinder radius.
C      kr        The wave number multiplied by the observation point radial
C                   distance from the center of the cylinder.
C      phi       The angular position of the observation point relative to
C                   the plane wave propagation vector.
C
C   The output arguments are:
C
C      n         The number of terms used in the summation.  If convergence
C                   is not reached in 50 terms, n is negative.
C      accl      The fractional accuracy at convergence (minimum 15 terms)
C                   or the fractional accuracy at the end of 50 terms.
C      dsp       The complex displacement field at the point (r, phi).
C
C   External Variables : none
C
C   Procedures called from this module :
C
C      jn        Bessel function of the first kind. (HP-UX utility.)
C      yn        Bessel function of the second kind. (HP-UX utility.)

$ALI
$ALIAS jn = 'jn' (%val, %val)
    AS yn = 'yn' (%val, %val)

      subroutine plane_scatter(acc, ka, kr, phi, n, accl, dsp)

      implicit none

      integer*4 n
      real*8 acc, ka, kr, accl, phi
      complex*16 dsp

      integer*4  j, 1
      real*8     acc2
      complex*16 aj, i, ij, x

      real*8     jn, yn, re, im
```

```
C
C  re = x * d/dx(jn(kx)).
C
C  im = x * d/dx(yn(kx)).
C
      re(n, kr) = n * jn(n, kr) - kr * jn((n+1), kr)
      im(n, kr) = n * yn(n, kr) - kr * yn((n+1), kr)

      acc2 = 0.
      i = dcmplx(0., 1.)

      if (ka .eq. 0.) then
         n = 0
         acc1 = 0.
         dsp = 0.
      else
C
C   j=0 term of the summation.
C
         j = 0
         aj = -re(j, ka) / dcmplx(re(j, ka), im(j, ka))
         dsp = aj * dcmplx(jn(j, kr), yn(j, kr))
C
C   Calculate the j=1 through j=n terms
C
         do j = 1, 50
            aj = re(j, ka) / dcmplx(re(j, ka), im(j, ka))
C
C      Calculate i^j.
C
            l = mod(j, 4)
            if (l .eq. 0) then
               ij = 1.
            else if (l .eq. 1) then
               ij = i
            else if (l .eq. 2) then
               ij = -1.
            else if (l .eq. 3) then
               ij = -i
            end if
            aj = -2. * ij * aj
C
C      Calculate the jth term of the sum.
C
            x = (aj * dcmplx(jn(j, kr), yn(j, kr))) * cos(j * phi)
C
C      Determine the accuracy level for the sum with j terms.
C
            acc1 = abs(x) / abs(dsp)
C
C      Add the jth term to the sum.
C
            dsp = dsp + x
            n = j
```

```
C
C      Test for convergence.
C
             if (j .ge. 15 .and. accl .le. acc
     1             .and. acc2 .le. acc) go to 10
             acc2 = accl
          end do
C
C   If convergence not reached, set n = -n.
C
          n = -n
   10     continue
       end if
       return

       end
```

H.2 Line Source Scattering

The program listed in this Section H.2 models the scattering of SH-wave pulses from a cylindrical cavity. The SH waves incident on the cavity radiate from a line source in a space that is represented either as an infinite lossless space or as a lossy two-layered half-space through the selection of appropriate options.

A sample input file and the first few lines of an output file for the program are shown in Section H.2.2.

H.2.1 Synthetic Seismogram Program

H.2.1.1 synseis

```
C
C    Procedure name: synseis
C
C    Author:         J.C. Biard
C
C    Discussion:     This is the main procedure for the program synseis.
C       This program calculates the time domain SH-wave seismic propagation
C       and scattering for a line source and an array of detectors in the
C       presence of an infinite cylindrical cavity of constant circular
C       cross-section in an elastic space.  The program is designed to allow
C       for the presence of a half-space of vacuum if desired, as well as a
C       surface layer located between the elastic half-space and the vacuum.
C       Attenuation and dispersion are also included.
C          The intent of the program is to produce data for a synthetic
C       seismogram.  As presently written, a set of time domain signals are
C       produced, one for each of a linear horizontal array of detectors.
C       These signals are the particle velocity amplitudes at each detector of
C       a horizontally polarized shear (SH) wave that has propagated through
C       the space and scattered from the cavity within it.  If selected, the
C       presence of the half-space surface and the surface layer produce
C       reflections of the direct propagating and scattered signals, which
C       become part of the time domain signal.  The inclusion of different
C       velocities, densities, and attenuation/dispersion values in the
C       different media allow for modeling a number of different earth
C       geometries.
C          The output data arrays containing the signal amplitudes at each
C       detector are written out to a file, which may then be interpreted
C       by a graphics presentation program to produce a synthetic seismogram.
C          The input data for the program are contained in a file, the name
C       of which is passed to the program on invocation.  An example input file
C       is shown below.
C          The output data are written to standard out, which must be
C       redirected to a file if the data are to be saved.
C
C    Invocation:     To run the program, after compilation and linking with the
C       subroutines, type "synseis infile", where infile is the input filename.
C       The output may be redirected to a file by typing the command as
C       "synseis infile > outfile", where outfile is the output file name.
```

```
C        The input file has the form:
C
C 3                    $ No. of reflections. (0=space, 1=1/2space, >=2=layered)
C 200., 1500.          $ Layer 1: SH-wave velocity(m/s), density(kg/m**3).
C 50., .05             $          Q factor, reference radial frequency.
C 2500., 2700.         $ Layer 2: SH-wave velocity(m/s), density(kg/m**3).
C 100., .05            $          Q factor, reference radial frequency.
C 1000., .9            $ Pulse peak freq., distribution factor.
C 39.78873577d0        $ Frequency step.
C 1., 5.               $ Tunnel radius, layer depth (h >= 0.).
C 0., -50.             $ Tunnel location (x, y).
C -52., -6.            $ Source location (x, y).
C -50., -6.            $ Detector array starting point (x, y). (y > -h)
C 51, 2.               $ No. of detectors, spacing.
C
C   Procedures called from this module:
C
C      inc_pulse    Subroutine that calculates the incident waveform
C                      for a source and detector in the lower region.
C      inc_pulse2   Subroutine that calculates the incident waveform
C                      for a source in the lower region and a detector
C                      on the surface of the half-space.
C      scat_pulse   Subroutine that calculates the scattered waveform
C                      for a source and detector in the lower region.
C      scat_pulse2  Subroutine that calculates the scattered waveform
C                      for a source in the lower region and a detector
C                      on the surface of the half-space.
C
C   Formal Arguments:
C
C      Input:
C         input     Filename of the input data file.
C
C      Output:
C         none
C
C   External variables:
C
C         nrefl     Number of reflection waves to calculate. (0<=nrefl<=6)
C         pi        Pi.
C         h         Thickness of the surface layer. (m)
C         c1        SH-wave velocity in the surface layer. (m / s)
C         rho1      Density of the surface layer. (kg / m)
C         c2        SH-wave velocity in the lower region. (m / s)
C         rho2      Density in the lower region. (kg / m)
C         a         Radius of the tunnel. (m)
C         alpha     Shape factor for the incident pulse.
C         w0        Peak radial frequency for the incident pulse.
C         df        Frequency step. (Hz)
C         xc        X position of the cavity center.
C         yc        Y position of the cavity center.
C         q1        Quality factor for the surface layer.
C         w1        Cutoff radial frequency for the surface layer.
C         q2        Quality factor for the lower region.
```

```
C        w2        Cutoff radial frequency for the lower region.
C
      program synseis(input)

      implicit none

      character input*80

      integer*4 nrefl
      real*8    pi, h, c1, rho1, c2, rho2, a, alpha, w0, df, xc, yc, q1,
     1          w1, q2, w2

      common /constants/ nrefl, pi, h, c1, rho1, c2, rho2, a,
     1                   alpha, w0, df, xc, yc, q1, w1, q2, w2

      integer*4 i, j, fdi, nx, np, time1, time2
      real*8    ti, ts, dt, scale, x, y, xs, ys, dx, x0, y0, alpha0, f0
      real*8    ipulse(4096), spulse(4096)

      save ipulse, spulse

      data fdi, np / 10, 4096/

      pi = 3.141592653589793d0
C
C    Obtain the run-time constants.
C
      open(unit = fdi, file = input, status = "OLD")
      read(fdi, *) nrefl
      read(fdi, *) c1, rho1
      read(fdi, *) q1, w1
      read(fdi, *) c2, rho2
      read(fdi, *) q2, w2
      read(fdi, *) f0, alpha0
      read(fdi, *) df
      read(fdi, *) a, h
      read(fdi, *) xc, yc
      read(fdi, *) xs, ys
      read(fdi, *) x0, y0
      read(fdi, *) nx, dx
      close(fdi)
      w0 = 2. * pi * f0
      alpha = alpha0 * w0 / pi
      dt = 1. / 256. / df
C
C    Find the 1-meter incident amplitude scale factor.
C
      call inc_pulse(0.d0, -500.d0, 1.d0, -500.d0, 1.d0, ti, ipulse)
      scale = 0.
      do i = 1, np
         scale = max(scale, abs(ipulse(i)))
      end do
      scale = 1. / scale
```

```
C
C   Write out the constants.
C
      write(6,*) nx, np, nrefl, h, c1, rho1, c2, rho2, w0, alpha0, df,
     1           dt, a, xc, yc, xs, ys, x0, y0, q1, w1, q2, w2, dx
C
C   Loop over the number of detector positions
C
      y = y0
      do i = 1, nx
         x = x0 + (i-1)*dx
C
C   Calculate the incident pulse.
C
         call time(time1)
         if (y0 .ne. 0) then
            call inc_pulse(xs, ys, x, y, scale, ti, ipulse)
         else
            call inc_pulse2(xs, ys, x, scale, ti, ipulse)
         end if
         call time(time2)
         time1 = time2 - time1
         write(7, *) time1
C
C   Calculate the scattered pulse.
C
         call time(time1)
         if (y0 .ne. 0) then
            call scat_pulse(xs, ys, x, y, scale, ts, spulse)
         else
            call scat_pulse2(xs, ys, x, scale, ts, spulse)
         end if
         call time(time2)
         time1 = time2 - time1
         write(7, *) time1
C
C   Write out the incident and scattered pulses to the output files.
C
         write(6, *) x, ti, ts
         write(6, *) (ipulse(j), j = 1, np)
         write(6, *) (spulse(j), j = 1, np)
      end do
      stop

      end
```

H.2.1.2 inc_pulse

```
C
C   Procedure name:  inc_pulse
C
C   Author         :  J.C. Biard
C
```

```
C   Discussion    : This procedure produces a numerical solution of
C         the equation for a line-source SH seismic wave propagating in
C         a dispersive absorptive half-space having a surface layer as given
C         by Equation (151) of Volume I.  The source and detection points are
C         located in the region below the surface layer.  At the detection
C         point, the time domain signal for the direct wave and up to nrefl
C         reflections from the layer interface and the half-space surface are
C         calculated using a high-frequency limit approximation formula.  This
C         result is reported in the array output.
C            To calculate a signal in the time domain, the frequency domain
C         transfer function is calculated and converted into the time domain
C         impulse response for the propagation path.  This is then convolved
C         with the initial signal to produce the signal at the detector.
C         This process is repeated for the direct wave and all allowed
C         reflections.  These time domain signals are then summed together
C         to present the final signal.
C            The transfer function that is calculated is multiplied by
C         j * omega before the Fourier transform in order to obtain the
C         time derivative of the displacement amplitude signal upon
C         convolving with the initial signal.  The resulting particle
C         velocity signal is the signal most often presented in seismic
C         studies.
C
C   Procedures called from this module:
C
C         fft2c     IMSL subroutine to calculate the Fourier transform.
C         jhc       subroutine to calculate Bessel and Hankel functions.
C         pulse     function to calculate the original signal.
C         zk        function to calculate the complex wavenumber.
C
C   Formal arguments:
C
C      Input:
C         xs        X position of the line source.
C         ys        Y position of the line source.
C         x         X position of the detector.
C         y         Y position of the detector.
C         scale     Scale factor to apply to the output waveform.
C
C      Output:
C         t         Time value of the initial element of the output array.
C         output    Array of the particle velocity signal waveform values.
C
C   External variables:
C
C         nrefl     Number of reflection waves to calculate. (0<=nrefl<=6)
C         pi        Pi.
C         h         Thickness of the surface layer. (m)
C         c1        SH-wave velocity in the surface layer. (m / s)
C         rho1      Density of the surface layer. (kg / m)
C         c2        SH-wave velocity in the lower region. (m / s)
C         rho2      Density in the lower region. (kg / m)
C         a         Radius of the cavity. (m)
          alpha     Shape factor for the incident pulse.
```

```
C          w0         Peak radial frequency for the incident pulse.
C          df         Frequency step. (Hz)
C          xc         X position of the cavity center.
C          yc         Y position of the cavity center.
C          q1         Quality factor for the surface layer.
C          w1         Cutoff radial frequency for the surface layer.
C          q2         Quality factor for the lower region.
C          w2         Cutoff radial frequency for the lower region.
C

      subroutine inc_pulse(xs, ys, x, y, scale, t, output)

      implicit none

      real*8     xs, ys, x, y, scale, t, output(4096)

      integer*4 nrefl
      real*8    pi, h, c1, rho1, c2, rho2, a, alpha, w0, df, xc, yc, q1,
     1          w1, q2, w2

      common /constants/ nrefl, pi, h, c1, rho1, c2, rho2, a, alpha,
     1                   w0, df, xc, yc, q1, w1, q2, w2

      integer*4  i, j, l, m, n, work(9), ier
      real*8     dk, k, r, dt, ti, sum, v, w, tsi2, u, tdy
      real*8     pwave(256), twave(256)
      complex*16 k2, kr, z, jn(2), hn(2), ftrans(256)
      complex*16 ij, krp, nu, rp, s, bn

      real*8     pulse
      complex*16 zk, znu

      znu(k) = zk(1, k * c2 / c1) / zk(2, k)

C
C   Initialize constants.
C
      data m / 8/

      ij = dcmplx(0.d0, 1.d0)
      dt = 1. / (256. * df)
      dk = 2. * pi * df / c2
      u = h / (abs(yc) - h)
      tsi2 = (rho1 * c1**2) / (rho2 * c2**2)
C
C   Calculate the path length time offset.
C
      r = sqrt((xs - x)**2 + (ys - y)**2)
      k = 128. * dk
      kr = zk(2,k) * r
      t = dble(kr) / k / c2 - 20.d0 * dt
```

```
C
C   Clear the output array.
C
      do i = 1, 4096
         output(i) = 0.d0
      end do
C
C   Calculate the array for the pulse waveform.
C
      do i = 1, 256
         ti = (i-1) * dt
         pwave(i) = pulse(ti)
      end do
C
C   Calculate the transfer function for the signal.
C      Loop over the direct wave (nrefl = 0) and all reflections.
C
      do i = 0, nrefl
         n = i - 1
C
C   Calculate the time delay for the signal.
C
         if (i .gt. 0) then
            k = 127. * dk
            nu = znu(k)
            rp = sqrt((x - xs)**2
     1              + (y + ys + 2.d0*(1.d0 - dble(n)*nu)*h)**2)
            tdy = dble(rp * zk(2, k)) / k / c2 - 20.d0 * dt
         else
            tdy = t
         end if
C
C   Calculate the value of the transfer function for the signal at
C      wavenumbers from 0. to 12.7.  The k = 0. value is set to 0.
C
         ftrans(1) = 0.d0
         do j = 2, 128
            k = (j - 1) * dk
C
C   Get the complex wavenumber.
C      This allows for attenuation and dispersion.
C
            k2 = zk(2, k)
            if (i .eq. 0) then
C
C   Find the direct term.
C
               kr = k2 * r
               call jhc(kr, 2, jn, hn)
               z = hn(1)
            else
```

```
C
C   Find a reflection term.
C
               nu = znu(k)
               rp = zsqrt((x - xs)**2
     1                + (y + ys + 2.d0*(1.d0 - dble(n)*nu)*h)**2)
C
C   Calculate the reflection coefficient.
C
               s = (tsi2 * nu - 1.d0) / (tsi2 * nu + 1.d0)
               if (n .eq. 0) then
                  bn = -s
               else
                  bn = (1.d0 - s**2) * s**(n-1)
     1               * zsqrt((1.d0 + u * dble(n) * nu)
     2                    / (1.d0 + u * dble(n) / nu))
               end if
               krp = k2 * rp
               call jhc(krp, 2, jn, hn)
               z = bn * hn(1)
            end if
            z = ij * pi * z * (dble(k2) / k / c2)**2
C
C   The procedure presently calculates the particle velocity signal.
C      To calculate displacement amplitudes, comment the next line.
C
            z = -ij * k * c2 * z
C
C   Remove the time delay from the transfer function.
C
            v = k * c2 * tdy
            ftrans(j) = zexp(-ij * v) * z
         end do
C
C   Force the high k values to zero.
C
         j = 5. / dk
         w = 128. * dk - 5.
         do l = j, 128
            v = l * dk - 5.
            ftrans(l) = ftrans(l) * (1. + cos(v * pi / w)) / 2.
         end do
C
C   Perform the Inverse Fourier transform.
C      Generate the full complex spectrum and conjugate it to prepare
C      for the back transform.
C
C   Reflect the spectrum for negative frequencies.
C
         do l = 2, 128
            ftrans(258-l) = conjg(ftrans(l))
         end do
         ftrans(129) = 0.
```

```
c
c   Conjugate the full spectrum.
c
          do i = 1, 256
             ftrans(i) = conjg(ftrans(i))
          end do
c
c   Fourier transform.
c
          call fft2c(ftrans, m, work)
c
c   Conjugate and normalize.
c
          do i = 1, 256
             ftrans(i) = conjg(ftrans(i)) * df
          end do
c
c   Convolve the pulse and the transfer function, and normalize.
c
          do i = 1, 256
             sum = 0.
             do j = 1, i
                sum = sum + pwave(i - j + 1) * dble(ftrans(j))
             end do
             twave(i) = scale * sum * dt / (4. * pi * rho2)
          end do
c
c   Find the offset of the time domain signal in the output array
c      and sum the signal into the total result.
c
          j = (tdy - t) / dt
          do i = 1, 256
             output(i + j) = output(i + j) + twave(i)
          end do
       end do
       return

       end
```

H.2.1.3 <u>inc_pulse2</u>

```
c
c   Procedure name:  inc_pulse2
c
c   Author          :  J.C. Biard
c
c   Discussion      :  This procedure produces a numerical solution of
c         the equation for a line-source SH seismic wave propagating in
c         a dispersive absorptive half-space having a surface layer as given by
c         Equation (167) in Volume I.  The source is located in the region below
c         the surface layer.  The detector point is located at the half-space
c         boundary, which is the top of the surface layer.  At the detection
```

```
C         point, the time domain signal for the direct wave and up to nrefl
C         reflections from the layer interface and the half-space surface are
C         calculated using a high-frequency limit approximation formula.
C         This result is reported in the array output.
C            To calculate a signal in the time domain, the frequency domain
C         transfer function is calculated and converted into the time domain
C         impulse response for the propagation path.  This is then convolved
C         with the initial signal to produce the signal at the detector.
C         This process is repeated for the direct wave and all allowed
C         reflections.  These time domain signals are then summed together
C         to present the final signal.
C            The transfer function that is calculated is multiplied by
C         j * omega before the Fourier transform in order to obtain the
C         time derivative of the displacement amplitude signal upon
C         convolving with the initial signal.  The resulting particle
C         velocity signal is the signal most often presented in seismic
C         studies.
C
C   Procedures called from this module:
C
C         fft2c      IMSL subroutine to calculate the Fourier transform.
C         jhc        subroutine to calculate Bessel and Hankel functions.
C         pulse      function to calculate the original signal.
C         zk         function to calculate the complex wavenumber.
C
C   Formal arguments:
C
C      Input:
C         xs         X position of the line source.
C         ys         Y position of the line source.
C         x          X position of the detector.
C         y          Y position of t he detector.
C         scale      Scale factor to apply to the output waveform.
C
C      Output:
C         t          Time value of the initial element of the output array.
C         output     Array of the particle velocity signal waveform values.
C
C   External variables:
C
C         nrefl      Number of reflection waves to calculate. (0<=nrefl<=6)
C         pi         Pi.
C         h          Thickness of the surface layer. (m)
C         c1         SH-wave velocity in the surface layer. (m / s)
C         rho1       Density of the surface layer. (kg / m)
C         c2         SH-wave velocity in the lower region. (m / s)
C         rho2       Density in the lower region. (kg / m)
C         a          Radius of the cavity. (m)
C         alpha      Shape factor for the incident pulse.
C         w0         Peak radial frequency for the incident pulse.
C         df         Frequency step. (Hz)
C         xc         X position of the cavity center.
C         yc         Y position of the cavity center.
C         q1         Quality factor for the surface layer.
```

```
C        w1        Cutoff radial frequency for the surface layer.
C        q2        Quality factor for the lower region.
C        w2        Cutoff radial frequency for the lower region.
C

      subroutine inc_pulse2(xs, ys, x, scale, t, output)

      implicit none

      real*8     xs, ys, x, scale, t, output(4096)

      integer*4 nrefl
      real*8     pi, h, cl, rhol, c2, rho2, a, alpha, w0, df, xc, yc, ql,
     1           wl, q2, w2

      common /constants/ nrefl, pi, h, cl, rhol, c2, rho2, a, alpha,
     1                   w0, df, sc, yc, ql, wl, q2, w2

      integer*4  i, j, l, m, n, work(9), ier
      real*8     dk, k, r, dt, ti, sum, v, w, tsi2
      real*8     u, tdy
      real*8     pwave(256), twave(256)
      complex*16 k2, kr, z, jn(2), hn(2), ftrans(256)
      complex*16 ij, krp, nu, bn, s, rp, p

      real*8     pulse
      complex*16 zk, znu

      znu(k) = zk(l, k * c2 / cl) / zk(2, k)
C
C    Initialize local constants.
C
      data m / 8/

      ij = dcmplx(0.d0, l.d0)
      dt = l. / (256. * df)
      dk = 2. * pi * df / c2
      u = h / (2.d0 * abs(yc) - h)
      tsi2 = (cl**2 * fhol) / (c2**2 * rho2)
C
C    Zero the output array.
C
      do i - 1, 4096
         output(i) - 0.d0
      end do
C
C    Calculate the array for the pulse waveform.
C
      do i = 1, 256
         ti = (i-1) * dt
         pwave(i) = pulse(ti)
      end do
```

```
C
C   Loop over the direct wave (0) and all reflections.
C
      do i = 0, nrefl
C
C   Calculate the path length time offset.
C
         k = 127. * dk
         nu = znu(k)
         rp = zsqrt((x - xs)**2
     1            + (ys + h - (1.d0 + 2.d0 * dble(i))*nu*h)**2)
         tdy = dble(rp * zk(2, k)) / k / c2 - 20.d0 * dt
         if (i .eq. 0) then
            t = tdy
         end if
C
C   Loop over the wavenumbers from dk to 127 * dk to get the transfer function
C      values.  Set the k = 0 term to zero.
C
         ftrans(1) = 0.d0
         do j = 2, 128
            k = (j - 1) * dk
            nu = znu(k)
C
C   Get the complex wavenumber.
C
            k2 = zk(2, k)
            krp = k2 * rp
C
C   Find the direct and reflection terms.
C
            rp = zsqrt((x - xs)**2
     1            + (ys + h - (1.d0 + 2.d0 * dble(i))*nu*h)**2)
C
C   Find the reflection coefficient.
C
            p = 4.d0 / (tsi2 * nu + 1.d0)
            s = (tsi2 * nu - 1.d0) / (tsi2 * nu + 1.d0)
            bn = p * s**i
     1         * zsqrt((1.d0 + u * (1.d0 + 2.d0 * dble(i)) * nu)
     2              / (1.d0 + u * (1.d0 + 2.d0 * dble(i)) / nu))
            call jhc(krp, 2, jn, hn)
            z = bn * hn(1)
            z = -ij * pi * z * (dble(k2) / k / c2)**2
C
C   The next statement generates the particle-velocity signal transfer
C      function.  To obtain displacement amplitude results, comment the
C      next line.
C
            z = z * ij * k * c2
```

```
C
C   Subtract the time delay.
C
            v = k * c2 * tdy
            ftrans(j) = zexp(-ij * v) * z
         end do
C
C   Force the high k values to zero.
C
         j = 5. / dk
         w = 128. * dk - 5.
         do l = j, 128
            v = l * dk - 5.
            ftrans(l) = ftrans(l) * (1. + cos(v * pi / w)) / 2.
         end do
C
C   Generate the full complex spectrum and conjugate it to prepare for the
C      back transform.
C
C   Reflect the spectrum for negative frequencies.
C
         do l = 2, 128
            trans(258-l) - conjg(ftrans(l))
         end do
         ftrans(129) = 0.
C
C   Conjugate the full spectrum.
C
         do l = 1, 256
            ftrans(l) = conjg(ftrans(l))
         end do
C
C   Inverse Fourier transform.
C
         call fft2c(ftrans, m, work)
C
C   Conjugate and normalize.
C
         do l = 1, 256
            ftrans(l) = conjg(ftrans(l)) * df
         end do
C
C   Convolve the pulse and the transfer function, and normalize.
C
         do l = 1, 256
            sum = 0.
            do j = 1, l
               sum = sum + pwave(l - j + 1) * dble(ftrans(j))
            end do
            twave(l) = scale * sum * dt / (4. * pi * rho2)
         end do
```

```
C
C   Find the offset into the output array, and add the contribution from
C      the present wave to the total result.
C
        j = (tdy - t) / dt
        do l = 1, 256
           output(l + j) = outpur(l + j) + twave(l)
        end do
      end do
      return

      end
```

H.2.1.4 jhc

```
C
C   Procedure Name:    jhc
C
C   Author:            J.C. Biard
C
C   Discussion:        This procedure calculates the complex Bessel and
C      Hankel functions for complex argument z for orders zero through
C      m-1 and places the results in the arrays jn and hn.
C         The results are obtained using series approximations to the
C      0th and 1st order Bessel and Neumann functions for complex argument
C      as given in Gradshteyn and Ryzhik (1965) and uses the recursion
C      formula for Bessel-class functions to obtain the values at the higher
C      orders.  There are two different series approximations used for the
C      calculations, one for small values of the argument and one for large
C      values of the argument.  The number of terms calculated and the
C      coefficients used were chosen so that there is a region of overlap
C      between the two solutions, thus providing an accurate and smooth
C      transition between the two regions.
C         The constants in the a array were precalculated to
C      speed the algorithm.
C
C   Procedures called from this module:
C
C      none
C
C   Formal arguments:
C
C      Input:
C
C         z             The complex valued argument for the functions.
C         m             The number of orders to find (from 0 to m-1).
C
C      Output:
C
C         jn            Array of the Bessel functions of order 0 to m-1 of z.
C                          Must be at least 2 elements.
```

```
C        hn              Array of the Hankel functions of order 0 to m-1 of z.
C                            Must be at least 2 elements.
C
C  External variables:
C
C     none
C

      subroutine jhc(z, m, jn, hn)

      implicit none

      integer*4  m
      complex*16 z, jn(m), hn(m)

      integer*4  i, j
      real*8     pi, c, a(100), a1(51), a2(49)
      complex*16 u, v, z2, z22, lnz2c, ji, ji1, ji2, ni, ni1, ni2
      complex*16 j1a, j2a, w, wp, wm, wf, u1, v1, aa, ab, im

      equivalence (a(1), a1(1)), (a(52), a2(1))

C
C  Initialize local constants.
C
      data pi, c/ 3.141592653589793d0, .5772156649015330d0/

      data a1/
     *    2.000000000000000d0, 3.000000000000000d0, 3.666666666666667d0,
     *    4.166666666666666d0, 4.566666666666666d0, 4.899999999999999d0,
     *    5.185714285714285d0, 5.435714285714285d0, 5.657936507936507d0,
     *    5.857936507936508d0, 6.039754689754689d0, 6.206421356421356d0,
     *    6.360267510267510d0, 6.503124653124654d0, 6.636457986457987d0,
     *    6.761457986457987d0, 6.879105045281516d0, 6.990216156392627d0,
     *    7.095479314287364d0, 7.195479314287364d0, 7.290717409525459d0,
     *    7.381626500434550d0, 7.468583022173680d0, 7.551916355507013d0,
     *    7.631916355507014d0, 7.708839432430090d0, 7.782913506504165d0,
     *    7.854342077932736d0, 7.923307595174115d0, 7.989974261840781d0,
     *    8.054490390873040d0, 8.116990390873040d0, 8.177596451479101d0,
     *    8.236419980890865d0, 8.293562533033722d0, 8.349118398589277d0,
     *    8.403172447643332d0, 8.455804026590700d0, 8.507086077872751d0,
     *    8.557086077872752d0, 8.605866565677630d0, 8.653485613296677d0,
     *    8.699997241203654d0, 8.745451786658199d0, 8.789896231102643d0,
     *    8.833374491972208d0, 8.875927683461569d0, 8.917594350128235d0,
     *    8.958410676658847d0, 8.998410676658846d0, 9.037626362933356d0/

      data a2/
     *    9.076087901394894d0, 9.113823750451498d0, 9.150860787488535d0,
     *    9.187224423852172d0, 9.222938709566458d0, 9.258026428864703d0,
     *    9.292509137485393d0, 9.326407492570139d0, 9.359740825903472d0,
     *    9.392527711149373d0, 9.424785775665503d0, 9.456531807411535d0,
     *    9.487781807411535d0, 9.518551038180766d0, 9.548854068483797d0,
     *    9.578704814752454d0, 9.608116579458336d0, 9.637102086704713d0,
     *    9.665673515276142d0, 9.693842529360650d0, 9.721620307138428d0,
     *    9.749017567412400d0, 9.776044594439426d0, 9.802711261106094d0,
     *    9.829027050579779d0, 9.855001076553805d0, 9.880642102194530d0,
     *    9.905958557891033d0, 9.930958557891033d0, 9.955649915915725d0,
     *    9.980040159818163d0,10.004136545360332d0,10.027946069169856d0,
     *   10.051475480934561d0,10.074731294888050d0,10.097719800635176d0,
     *   10.120447073362449d0,10.142918983474809d0,10.165141205697031d0,
     *   10.187119227675053d0,10.208858358109836d0,10.230363734453922d0,
     *   10.251640330198603d0,10.272692961777549d0,10.293526295110883d0,
     *   10.314144851811914d0,10.334553015077221d0,10.354755035279242d0,
     *   10.374755035279241d0/

      im = dcmplx(0.d0, 1.d0)
C
C   If abs(z) is less than 15 then use the small argument approximation.
C      First calculate the Bessel and Neumann function values for z.
C
      if (abs(z) .lt. 15.) then
         z2 = z / 2.d0
         lnz2c = 2.d0 * (zlog(z2) + c)
         z22 = z2*z2
         ji2 = 0.d0
         ji1 = 0.d0
         ni2 = 0.d0
         ni1 = 0.d0
C
C   Sum the terms for the series, starting with the highest index.
C
         do i = 30, 1, -1
            u = -z22 / dble(i) / dble(i)
            v = -z22 / dble(i) / dble(i+1)
            aa = lnz2c - a(i)
            ab = lnz2c - a(i) - 1.d0 / dble(i+1)
            ji2 = u * (1.d0 + ji2)
            ji1 = v * (1.d0 + ji1)
            ni2 = u * (aa + ni2)
            ni1 = v * (ab + ni1)
         end do
C
C   Add the zeroth terms to the sums.
C
         ji2 = ji2 + 1.d0
         ji1 = z2 * (1.d0 + ji1)
         ni2 = (lnz2c + ni2) / pi
         ni1 = (z2 * (lnz2c - 1.d0 + ni1) - 1.d0 / z2) / pi
C
C   Generate the Hankel function values from the Bessel and Neumann.
C
         ni2 = ji2 + im * ni2
         ni1 = ji1 + im * ni1
      else
```

```
C
C   If abs(z) is greater than 15 then use the large argument approx.
C
        z2 = 8.d0 * z
        z22 = z2**2
        u = 1.
        ul = 1.
        v = 1.
        vl = 1.
        ji2 = 1.
        j2a = 1.
        jil = 1.
        jla = 3.
C
C   Sum the terms of the series.
C
        do i = 1, 10
           w = (4.d0 * dble(i) - 1.d0)
           wp = w + 2.d0
           wm = w - 2.d0
           wf = 2.d0 * dble(i)
           u = -(w * wm)**2 / (wf * (wf - 1.d0) * z22) * u
           v = wp * u / w
           wm = w
           w = wp
           wp = w + 2.d0
           ul = -(w * wm)**2 / (wf * (wf + 1.d0) * z22) * ul
           vl = wp * ul / w
           ji2 = u + ji2
           j2a = ul + j2a
           jil = -v + jil
           jla = vl + jla
        end do
C
C   Apply the external multiplying factors, and generate the Bessel and
C      Hankel function values.
C
        u = z - pi / 4.d0
        v = zsqrt(2.d0 / pi / z)
        ni2 = v * (ji2 - im * j2a / z2) * zexp(im * u)
        ji2 = v * (ji2*zcos(u) + j2a*zsin(u)/z2)
        nil = v * (jla / z2 - im * jil) * zexp(im * u)
        jil = v * (jil*zsin(u) + jla*zcos(u)/z2)
     end if
C
C   Store the 0th and 1st order Bessel and Hankel function values.
C
     jn(1) = ji2
     jn(2) = jil
     hn(1) = ni2
     hn(2) = nil
```

```
C
C   Apply the recursion relation to obtain the higher order function values.
C
      do i = 3, m
         ji = 2.d0 * dble(i - 2) / z * ji1 - ji2
         ji2 = ji1
         ji1 = ji
         jn(i) = ji
         ni = 2.d0 * dble(i - 2) / z * ni1 - ni2
         ni2 = ni1
         ni1 = ni
         hn(i) = ni
      end do
      return

      end
```

H.2.1.5 Function pulse

```
C
C   Procedure name:   pulse
C
C   Author:           J.C. Biard
C
C   Discussion:       This procedure calculates the value of a time-domain
C      wave packet function at the time, t, using Equation (63) in Volume I.
C      The shape and central frequency of the pulse is determined by the
C      external variables alpha and w0.  The variable alpha controls the
C      width of the pulse envelope and the variable w0 sets the central
C      frequency.  Alpha is considered as a fraction of w0 divided by pi.
C
C   Procedures called from this module:
C
C      none
C
C   Formal arguments:
C
C      Input:
C
C         t          The time in seconds for which the function value is
C                       desired.
C
C      Output:
C
C         pulse      The value of the pulse function at time t.
C
C   External variables:
C
C         nrefl      Number of reflection waves to calculate. (0<=nrefl<=6)
C         pi         Pi.
C         h          Thickness of the surface layer. (m)
C         c1         SH-wave velocity in the surface layer. (m / s)
```

```
C         rho1      Density of the surface layer. (kg / m)
C         c2        SH-wave velocity in the lower region. (m / s)
C         rho2      Density in the lower region. (kg / m)
C         a         Radius of the cavity. (m)
C         alpha     Shape factor for the incident pulse.
C         w0        Peak radial frequency for the incident pulse.
C         df        Frequency step. (Hz)
C         xc        X position of the cavity center.
C         yc        Y position of the cavity center.
C         q1        Quality factor for the surface layer.
C         w1        Cutoff radial frequency for the surface layer.
C         q2        Quality factor for the lower region.
C         w2        Cutoff radial frequency for the lower region.
C

      real*8 function pulse(t)

      implicit none

      real*8 t

      integer*4 nrefl
      real*8    pi, h, c1, rho1, c2, rho2, a, alpha, w0, df, xc, yc, q1,
     1          w1, q2, w2

      common /constants/ nrefl, pi, h, c1, rho1, c2, rho2, a, alpha,
     1                   w0, df, xc, yc, q1, w1, q2, w2

      real*8 at, w0t

C
C   Calculate the function value.
C
      at = alpha * t
      w0t = w0 * t
      pulse = at * exp(-at) * sin(w0t)
      return

      end
```

H.2.1.6 scat_pulse

```
C
C   Procedure name:  scat_pulse
C
C   Author         :  J.C. Biard
C
C   Discussion     :  This procedure produces a numerical solution of
C        the equation for a line-source SH seismic wave propagating in
C        a dispersive absorptive half-space having a surface layer as
C        expressed by Equation (148a) in Volume I.  The wave scatters from
C        an infinite cylindrical cavity of radius, a, located yc meters below
```

```
C        the half-space surface.  The source and detection points are located
C        in the region below the surface layer.  At the detection point the
C        time domain signal for the scattered wave, including nrefl pre-
C        scattering and nrefl post-scattering reflections from the surface
C        layer boundary and the half-space boundary, is calculated using a
C        high-frequency limit approximation formula.  The results are reported
C        in the array wave.
C           To calculate a signal in the time domain, the frequency domain
C        transfer function is calculated and converted into the time domain
C        impulse response for the propagation path.  This is then convolved
C        with the initial signal to produce the signal at the detector.
C        This process is repeated for the direct wave and all allowed
C        reflections.  These time domain signals are then summed together
C        to present the final signal.
C           The transfer function that is calculated is multiplied by
C        j * omega before the Fourier transform in order to obtain the
C        time derivative of the displacement amplitude signal upon
C        convolving with the initial signal.  The resulting particle
C        velocity signal is the signal most often presented in seismic
C        studies.
C
C  Procedures called from this module:
C
C        fft2c     IMSL subroutine to calculate the Fourier transform.
C        jhc       subroutine to calculate Bessel and Hankel functions.
C        pulse     function to calculate the original signal.
C        zk        function to calculate the complex wavenumber.
C
C  Formal arguments:
C
C     Input:
C        xs        X position of the line source.
C        ys        Y position of the line source.
C        x         X position of the detector.
C        y         Y position of the detector.
C        scale     Scale factor to apply to the output waveform.
C
C     Output:
C        t         Time value of the initial element of the output array.
C        wave      Array of the particle velocity signal waveform values.
C
C  External variables:
C
C        nrefl     Number of reflection waves to calculate. (0<=nrefl<=6)
C        pi        Pi.
C        h         Thickness of the surface layer. (m)
C        c1        SH-wave velocity in the surface layer. (m / s)
C        rho1      Density of the surface layer. (kg / m)
C        c2        SH-wave velocity in the lower region. (m / s)
C        rho2      Density in the lower region. (kg / m)
C        a         Radius of the cavity. (m)
C        alpha     Shape factor for the incident pulse.
C        w0        Peak radial frequency for the incident pulse.
C        df        Frequency step. (Hz)
```

```
C        xc        X position of the cavity center.
C        yc        Y position of the cavity center.
C        q1        Quality factor for the surface layer.
C        w1        Cutoff radial frequency for the surface layer.
C        q2        Quality factor for the lower region.
C        w2        Cutoff radial frequency for the lower region.
C        hnkrs     Array of variables made external to save memory.
C        hnkr     Array of variables made external to save memory.
C

      subroutine scat_pulse(xs, ys, x, y, scale, t, wave)

      implicit none

      real*8    xs, ys, x, y, scale, t, wave(4096)

      integer*4 nrefl
      real*8    pi, h, c1, rho1, c2, rho2, a, alpha, w0, df, xc, yc, q1,
     1          w1, q2, w2

      common /constants/ nrefl, pi, h, c1, rho1, c2, rho2, a,
     1                   alpha, w0, df, xc, yc, q1, w1, q2, w2

      integer*4  i, n, j, l, nr, nrs, virgin, ier, m, work(9)
      real*8     dt, dk, sum, k, tdy, c, cs, phis, phi, theta
      real*8     acc, acc1, w, v, u, tsi2, acc2
      real*8     pwave(256), twave(256)
      complex*16 k2, z, dispi, ij, aj, ka, krs, kr, nu, r, rs, cnr, cnrs
      complex*16 s, ss, b
      complex*16 trndsp(128), spect(256), hnka(0:50)
      complex*16 hnkrs(0:49,2:128,0:5), hnkr(0:49,2:128,0:5)
      complex*16 jnka(0:50), jpka(0:49,2:128), hpka(0:49,2:128)

      common /hankel/ hnkrs, hnkr

      save jpka, hpka

      real*8     pulse, zatan
      complex*16 zk, znu

      equivalence (trndsp(1), spect(1))

      znu(k) = zk(1, k * c2 / c1) / zk(2, k)
      zatan(s, x) = .5 * (datan2(dble(s), (x + dimag(s)))
     1                  + datan2(dble(s), (x - dimag(s))))
C
C    Initialize local constants
C

      data virgin / 0/
      data m / 8/
      data acc / .0001/
      ij = dcmplx(0., 1.)
      dt = 1. / (256. * df)
      dk = df * 2 * pi / c2
      u = h / (abs(yc) - h)
      c = x - xc
      cs = xs - xc
      tsi2 = (c1**2 * rho1) / (c2**2 * rho2)
C
C   Zero the output array.
C
      do i = 1, 4096
         wave(i) = 0.
      end do
C
C   Generate the pulse waveform.
C
      do l = 1, 256
         v = dble(l - 1) * dt
         pwave(l) = pulse(v)
      end do
C
C   Calculate the initial path length time offset.
C
      k = 128. * dk
      r = sqrt((x - xc)**2 + (y - yc)**2)
      rs = sqrt((xs - xc)**2 + (ys - yc)**2)
      t = dble((rs + r) * zk(2, k)) / k / c2 - 20. * dt

C
C   Get the scattering coefficients for the cavity.
C
      if (virgin .eq. 0) then
         virgin = 1
         do i = 2, 128
            k = dble(i-1) * dk
            k2 = zk(2, k)
            ka = k2 * a
            call jhc(ka, 51, jnka, hnka)
            do j = 0, 49
               jpka(j,i) = dble(j) * jnka(j) - ka * jnka(j+1)
               hpka(j,i) = dble(j) * hnka(j) - ka * hnka(j+1)
            end do
         end do
      end if
C
C   Calculate the Hankel functions used in the field equation.
C      This is done to prevent recalculation later.
C
C
C   Loop over the direct term and the number of reflections.
C
      do nrs = 0, nrefl
```

```
C
C   Get the direct source-scatterer and detector-scatterer distances.
C
         if (nrs .eq. 0) then
            ss = ys - yc
            rs = sqrt(cs**2 + ss**2)
            s = y - yc
            r = sqrt(c**2 + s**2)
         end if
C
C   Loop over the wavenumber from dk to 127 * dk.
C
         do i = 2, 128
            k = dble(i-1) * dk
            k2 = zk(2, k)
C
C   Get the reflected source-scatterer and detector-scatterer distances.
C
            if (nrs .gt. 0) then
               nu = znu(k)
               ss = -(ys + yc + 2.d0 * (1.d0 - dble(nrs-1) * nu) * h)
               rs = zsqrt(cs**2 + ss**2)
               s = -(y + yc + 2.d0 * (1.d0 - dble(nrs-1) * nu) * h)
               r = zsqrt(c**2 + s**2)
            end if
            krs = k2 * rs
            kr = k2 * r
C
C  Calculate the Hankel function values.
C
            call jhc(krs, 50, jnka, hnkrs(0,i,nrs))
            call jhc(kr, 50, jnka, hnkr(0,i,nrs))
         end do
      end do
C
C   Begin the Solution.
C      Loop over the direct and all the reflection terms for the source.
C
      do nrs = 0, nrefl
C
C   Get the direct source-scatterer distance and angle.
C      Set the source-scatterer amplitude factor.
C
         if (nrs .eq. 0) then
            ss = ys - yc
            rs = sqrt(cs**2 + ss**2)
            phis = zatan(ss, cs)
            cnrs = 1.d0
         end if
C
C      Loop over the direct and all the reflection terms for the detector.
C
         do nr = 0, nrefl
```

```
C
C   Get the direct detector-scatterer distance and angle.
C      Set the detector-scatterer amplitude factor.
C
            if (nr .eq. 0) then
               s = y - yc
               r = sqrt(c**2 + s**2)
               phi = zatan(s, c)
               cnr = 1.d0
            end if
C
C   Loop over the wavenumber values from 0. to 127 * dk
C
            do i = 1, 128
               k = dble(i - 1) * dk
               k2 = zk(2, k)
               if (i .eq. 1) then
                  nu = c2 / c1
               else
                  nu = znu(k)
               end if
               b = (tsi2 * nu - 1.d0) / (tsi2 * nu + 1.d0)
C
C   Get the reflected source-scatterer distance and angle.
C      Set the source-scatterer amplitude factor.
C
               if (nrs .gt. 0) then
                  ss = -(ys + yc + 2.d0 * (1.d0 - dble(nrs-1) * nu) * h)
                  rs = zsqrt(cs**2 + ss**2)
                  phis = zatan(ss, cs)
                  if (nrs .eq. 1) then
                     cnrs = -b
                  else
                     cnrs = b**(nrs-2) * (1.d0 - b**2)
     1                    * zsqrt((1.d0+u*nu*dble(nrs-1))
     2                          / (1.d0+u*dble(nrs-1)/nu))
                  end if
               end if
C
C   Get the reflected detector-scatterer distance and angle.
C      Set the detector-scatterer amplitude factor.
C
               if (nr .gt. 0) then
                  s = -(y + yc + 2.d0 * (1.d0 - dble(nr-1) * nu) * h)
                  r = zsqrt(c**2 + s**2)
                  phi = zatan(s, c)
                  if (nr .eq. 1) then
                     cnr = -b
                  else
                     cnr = b**(nr-2) * (1.d0 - b**2)
     1                   * zsqrt((1.d0+u*nu*dble(nr-1))
     2                        / (1.d0+u*dble(nr-1)/nu))
                  end if
               end if
```

```
C
C   Get scattered field value.
C
               krs = k2 * rs
               kr = k2 * r
               theta = phis - phi
C
C   Do the scattered field transfer function summation.
C      j=0 term of the summation.
C
               j = 0
               if (k .eq. 0.) then
                  dispi = 0.
               else
                  aj = jpka(j,i) / hpka(j,i)
     1               * hnkrs(j,i,nrs) * hnkr(j,i,nr)
                  dispi = aj
               end if
C
C   Calculate the j=1 through j=n terms
C
               acc2 = 0.
               do j = 1, 49
C
C      Calculate the jth term of the sum.
C
                  if (k .eq. 0.) then
                     z = -ij * cos(dble(j) * theta)
     1                 / (rs * r / a**2)**j / dble(j)
                  else
                     aj = 2.d0 * jpka(j,i) / hpka(j,i)
     1                  * hnkrs(j,i,nrs) * hnkr(j,i,nr)
                     z = aj * cos(dble(j) * theta)
                  end if
C
C      Add the jth term to the sum.
C
                  dispi = dispi + z
C
C      Determine the accuracy level for the sum with j terms.
C
                  if (abs(dispi) .eq. 0.) then
                     acc1 = acc
                  else
                     acc1 = abs(z) / abs(dispi)
                  end if
C
C           Test for convergence

                  n = j
                  if (acc1 .le. acc .and. acc2 .le. acc
     1                              .and. j .ge. 10) go to 10

                  acc2 = acc1
               end do
```

```
C
C   Flag non-convergence error.
C
               write(7, 2000) acc, accl
   10          continue
               dispi = -ij * pi * cnr * cnrs * dispi
C
C   The next statement is for calculating particle velocity seismograms.
C      To calculate displacement seismograms, comment the statement.
C
               dispi = -ij * dispi * c2 * k
C
C   Apply partial normalization and add the term to the transfer function.
C
               if (k .eq. 0.) then
                  dispi = dispi / c2**2
               else
                  dispi = dispi * (dble(k2) / k / c2)**2
               end if
               trndsp(i) = dispi
            end do
C
C   Remove path length time delay from the transfer function.
C
            k = 127.d0 * dk
            k2 = zk(2, k)
            tdy = dble((rs + r) * k2) / k / c2 - 20. * dt
            v = dk * c2 * tdy
            do i = 1, 128
               k = dble(i - 1) * v
               trndsp(i) = trndsp(i) * zexp(-ij * k)
            end do
C
C   Force the high k values to zero.
C
            j = 5. / dk
            w = 128. * dk - 5.
            do l = j, 128
               v = l * dk - 5.
               trndsp(l) = trndsp(l) * (1. + cos(v * pi / w)) / 2.
            end do
C
C   Generate the full complex spectrum and conjugate it to prepare for the
C      back transform.
C
C   Reflect the spectrum for negative frequencies.
C
            do l = 2, 128
               spect(258-l) = conjg(trndsp(l))
            end do
            spect(129) = 0.
C
C   Conjugate the full spectrum.
C
```

```
              do l = 1, 256
                 spect(l) = conjg(spect(l))
              end do
C
C   Inverse Fourier transform.
C
              call fft2c(spect, m, work)
C
C   Conjugate and normalize.
C
              do l = 1, 256
                 spect(l) = df * conjg(spect(l))
              end do
C
C   Convolve the pulse and the transfer function, and normalize.
C
              do l = 1, 256
                 sum = 0.
                 do j = 1, l
                    sum = sum + pwave(l - j + 1) * dble(spect(j))
                 end do
                 twave(l) = scale * sum * dt / (4. * pi * rho2)
              end do
C
C   Find the time offset and add the pulse to the time domain signal.
C
              i = (tdy - t) / dt
              do l = 1, 256
                 if (l + i .ge. 1) then
                    wave(l + i) = wave(l + i) + twave(l)
                 end if
              end do
           end do
        end do
        return

 2000   format("Convergence failed.  Limit = ",g14.6,", value = ",g14.6)

        end
```

H.2.1.7 scat_pulse2

```
C
C   Procedure name:  scat_pulse2
C
C   Author         :  J.C. Biard
C
C   Discussion     :  This procedure produces a numerical solution of
C         the equation for a line-source SH seismic wave propagating in
C         a dispersive absorptive half-space having a surface layer as
C         expressed by Equation (165) in Volume I.  The wave scatters from
```

```
C           an infinite cylindrical void of radius a located yc meters below
C           the half-space surface.  The source point is located in the region
C           below the surface layer and the detector point is located at the
C           surface of the the half-space.  At the detection point the time
C           domain signal for the scattered wave, including nrefl pre-
C           scattering and nrefl post-scattering reflections from the surface
C           layer boundary and the half-space boundary, is calculated using a
C           high-frequency limit approximation formula.  The results are
C           reported in the array wave.
C              To calculate a signal in the time domain, the frequency domain
C           transfer function is calculated and converted into the time domain
C           impulse response for the propagation path.  This is then convolved
C           with the initial signal to produce the signal at the detector.
C           This process is repeated for the direct wave and all allowed
C           reflections.  These time domain signals are then summed together
C           to present the final signal.
C              The transfer function that is calculated is multiplied by
C           j * omega before the Fourier transform in order to obtain the
C           time derivative of the displacement amplitude signal upon
C           convolving with the initial signal.  The resulting particle
C           velocity signal is the signal most often presented in seismic
C           studies.
C
C     Procedures called from this module:
C
C           fft2c     IMSL subroutine to calculate the Fourier transform.
C           jhc       subroutine to calculate Bessel and Hankel functions.
C           pulse     function to calculate the original signal.
C           zk        function to calculate the complex wavenumber.
C
C     Formal arguments:
C
C        Input:
C           xs        X position of the line source.
C           ys        Y position of the line source.
C           x         X position of the detector.
C           scale     Scale factor to apply to the output waveform.
C
C        Output:
C           t         Time value of the initial element of the output array.
C           wave      Array of the particle velocity signal waveform values.
C
C     External variables:
C
C           nrefl     Number of reflection waves to calculate. (0<=nrefl<=6)
C           pi        Pi.
C           h         Thickness of the surface layer. (m)
C           c1        SH-wave velocity in the surface layer. (m / s)
C           rho1      Density of the surface layer. (kg / m)
C           c2        SH-wave velocity in the lower region. (m / s)
C           rho2      Density in the lower region. (kg / m)
C           a         Radius of the cavity. (m)
C           alpha     Shape factor for the incident pulse.
C           w0        Peak radial frequency for the incident pulse.
```

```
C        df        Frequency step. (Hz)
C        xc        X position of the cavity center.
C        yc        Y position of the cavity center.
C        q1        Quality factor for the surface layer.
C        w1        Cutoff radial frequency for the surface layer.
C        q2        Quality factor for the lower region.
C        w2        Cutoff radial frequency for the lower region.
C        hnkrs     Array of variables made external to save memory.
C        hnkr      Array of variables made external to save memory.
C

      subroutine scat_pulse2(xs, ys, x, scale, t, wave)

      implicit none

      real*8    xs, ys, x, scale, t, wave(4096)

      integer*4 nrefl
      real*8    pi, h, c1, rho1, c2, rho2, a, alpha, w0, df, xc, yc, q1,
     1          w1, q2, w2

      common /constants/ nrefl, pi, h, c1, rho1, c2, rho2, a,
     1                   alpha, w0, df, xc, yc, q1, w1, q2, w2

      integer*4  i, n, j, l, nr, nrs, virgin, ier, m, work(9)
      real*8     dt, dk, sum, k, tdy, c, cs
      real*8     acc, acc1, w, v, u, tsi2
      real*8     acc2, us, phi, phis, theta
      real*8     pwave(256), twave(256)
      complex*16 nu, r, rs, s, ss, cnr, cnrs
      complex*16 k2, z, dispi, ij, aj, ka, krs, kr, b
      complex*16 trndsp(128), spect(256), hnka(0:50)
      complex*16 hnkrs(0:49,2:128,0:5), hnkr(0:49,2:128,0:5)
      complex*16 jnka(0:50), jpka(0:49,2:128), hpka(0:49,2:128)

      common /hankel/ hnkrs, hnkr

      save jpka, hpka

      equivalence (trndsp(1), spect(1))

      real*8     pulse, zatan
      complex*16 zk, znu

      znu(k) = zk(1, k * c2 / c1) / zk(2, k)
      zatan(s, x) = .5 * (datan2(dble(s), (x + dimag(s)))
     1              + datan2(dble(s), (x - dimag(s))))
C
C     Initialize local constants.
C
      data virgin / 0/
      data m / 8/
      data acc / .0001/
      ij = dcmplx(0., 1.)
      dt = 1. / (256. * df)
      dk = df * 2 * pi / c2
      us = h / (abs(yc) - h)
      u = h / (2.d0 * abs(yc) - h)
      c = x - xc
      cs = xs - xc
      tsi2 = (c1**2 * rho1) / (c2**2 * rho2)
C
C   Zero the output array.
C
      do i = 1, 4096
         wave(i) = 0.
      end do
C
C   Generate the pulse waveform.
C
      do l = 1, 256
         v = dble(l - 1) * dt
         pwave(l) = pulse(v)
      end do
C
C   Get the scattering coefficients for the tunnel.
C
C   If this is the first call to the subroutine, then calculate the coefficients
C
      if (virgin .eq. 0) then
         virgin = 1
C
C   Loop over wavenumbers from dk to 127. * dk.
C
         do i = 2, 128
            k = dble(i-1) * dk
C
C   Get the complex wave number.
C
            k2 = zk(2, k)
            ka = k2 * a
C
C   Find the values of the Bessel and Hankel functions for the
C      argument ka.
C
            call jhc(ka, 51, jnka, hnka)
C
C   Calculate the coefficients from the Bessel and Hankel functions.
C
            do j = 0, 49
               jpka(j,i) = dble(j) * jnka(j) - ka * jnka(j+1)
               hpka(j,i) = dble(j) * hnka(j) - ka * hnka(j+1)
            end do
         end do
      end if
```

```
C
C   Calculate the Hankel function values used in the solution for
C      the scattered field.  This is done in advance to save execution
C      time.
C
C   Loop over the direct and all image terms.
C
      do nrs = 0, nrefl
C
C   Get the direct source-scatterer distance.
C
         if (nrs .eq. 0) then
            ss = ys - yc
            rs = zsqrt(cs**2 + ss**2)
         end if
C
C   Loop over wavenumber values from dk to 127. * dk.
C
         do i = 2, 128
            k = dble(i-1) * dk
C
C   Get the complex wavenumber.
C
            k2 = zk(2, k)
            nu = znu(k)
C
C   Calculate the source-scatterer distance with nrs reflections.
C
            if (nrs .gt. 0) then
               ss = -(ys + yc + 2.d0 * (1.d0 - dble(nrs-1) * nu) * h)
               rs = zsqrt(cs**2 + ss**2)
            end if
C
C   Calculate the detector-scatterer distance with nrs reflections.
C
            s = -(h + yc - (1.d0 + 2.d0 * dble(nrs)) * nu * h)
            r = zsqrt(c**2 + s**2)
            krs = k2 * rs
            kr = k2 * r
C
C   Find the Hankel function values for the arguments krs and kr.
C
            call jhc(krs, 50, jnka, hnkrs(0,i,nrs))
            call jhc(kr, 50, jnka, hnkr(0,i,nrs))
         end do
      end do
C
C   Begin the solution.
C
C   Loop over all the direct and reflection terms for the source.
C
      do nrs = 0, nrefl
```

```
C
C   Calculate the direct source-scatterer distance, angle, and amplitude factor.
C
         if (nrs .eq. 0) then
            ss = ys - yc
            rs = zsqrt(cs**2 + ss**2)
            phis = zatan(ss, cs)
            cnrs = 1.d0
         end if
C
C   Loop over all the direct and reflection terms for the detector.
C
         do nr = 0, nrefl
C
C   Calculate values for 0. <= ka <= (nk - 1)*dka, stepping by dka.
C
            do i = 1, 128
               k = dble(i - 1) * dk
               k2 = zk(2, k)
               if (k .eq. 0) then
                  nu = c2 / c1
               else
                  nu = znu(k)
               end if
               b = (tsi2 * nu - 1.d0) / (tsi2 * nu + 1.d0)
C
C   Calculate the source-scatterer distance, angle, and amplitude factor
C      with nrs reflections.
C
               if (nrs .gt. 0) then
                  ss = -(ys + yc + 2.d0 * (1.d0 - dble(nrs-1) * nu) * h)
                  rs = zsqrt(cs**2 + ss**2)
                  phis = zatan(ss, cs)
                  if (nrs .eq. 1) then
                     cnrs = -b
                  else
                     cnrs = b**(nrs-2) * (1.d0 - b**2)
     1                    * zsqrt((1.d0+us*nu*dble(nrs-1))
     2                         / (1.d0+us*dble(nrs-1)/nu))
                  end if
               end if
C
C   Calculate the detector-scatterer distance, angle, and amplitude factor
C      with nr reflections.
C
               s = -(h + yc - (1.d0 + 2.d0 * dble(nr)) * nu * h)
               r = zsqrt(c**2 + s**2)
               phi = zatan(s, c)
               cnr = b**nr * 4.d0 / (tsi2 * nu + 1.d0)
     1            * zsqrt((1.d0+u*nu*(1.d0 + 2.d0 * dble(nr)))
     2                 / (1.d0+u*(1.d0 + 2.d0 * dble(nr))/nu))
```

```
C
C   Get scattered field value.
C
            krs = k2 * rs
            kr = k2 * r
            theta = phis - phi
C
C   Begin summation of the scattered field terms.
C
C   j=0 term of the summation
C
            j = 0
            if (k .eq. 0.) then
               dispi = 0.
            else
               aj = jpka(j,i) / hpka(j,i)
  1               * hnkrs(j,i,nrs) * hnkr(j,i,nr)
               dispi = aj
            end if
C
C   Calculate the j=1 through j=n terms.
C
            acc2 = 0.
            do j = 1, 49
C
C      Calculate the jth term of the sum.
C
               if (k .eq. 0.) then
                  z = -1j * cos(dble(j) * theta)
  1                  / (rs * r / a**2)**j / dble(j)
               else
                  aj = 2.d0 * jpka(j,i) / hpka(j,i)
  1                    * hnkrs(j,i,nrs) * hnkr(j,i,nr)
                  z = aj * cos(dble(j) * theta)
               end if
C
C      Add the jth term to the sum.
C
               dispi = dispi + z
C
C      Determine the accuracy level for the sum with j terms
C
               if (abs(dispi) .eq. 0.) then
                  acc1 = acc
               else
                  acc1 = abs(z) / abs(dispi)
               end if
C
C           Test for convergence.
C
               n = j
               if (acc1 .le. acc .and. acc2 .le. acc
  1                                 .and. j .ge. 10) go to 10
```

```
                  acc2 = acc1
               end do
C
C   Flag non-convergence error.
C
               write(7, 2000) acc, acc1
   10          continue
               dispi = -ij * pi * cnr * cnrs * dispi
C
C   The next statement is for calculating particle velocity seismograms.
C      To calculate displacement seismograms, comment the statement.
C
               dispi = -ij * dispi * c2 * k
C
C   Apply partial normalization and add the term to the transfer function array.
C
               if (k .eq. 0.) then
                  dispi = dispi / c2**2
               else
                  dispi = dispi * (dble(k2) / k / c2)**2
               end if
               trndsp(i) = dispi
            end do
C
C   Remove path length time delay from the transfer function.
C
            k = 127.d0 * dk
            k2 = zk(2, k)
            tdy = dble((rs + r) * k2) / k / c2 - 20. * dt
            if (nr .eq. 0 .and. nrs .eq. 0) then
               t = tdy
            end if
            v = dk * c2 * tdy
            do i = 1, 128
               k = dble(i - 1) * v
               trndsp(i) = trndsp(i) * zexp(-ij * k)
            end do
C
C   Force the high k values to zero.
C
            j = 5. / dk
            w = 128. * dk - 5.
            do l = j, 128
               v = l * dk - 5.
               trndsp(l) = trndsp(l) * (1. + cos(v * pi / w)) / 2.
            end do
C
C   Generate the full complex spectrum and conjugate it to prepare for the
C      back transform.
C
```

```
C   Reflect the spectrum for negative frequencies.
C
            do l = 2, 128
               spect(258-l) = conjg(trndsp(l))
            end do
            spect(129) = 0.
C
C   Conjugate the full spectrum.
C
            do l = 1, 256
               spect(l) = conjg(spect(l))
            end do
C
C   Inverse Fourier transform.
C
            call fft2c(spect, m, work)
C
C   Conjugate and normalize.
C
            do l = 1, 256
               spect(l) = df * conjg(spect(l))
            end do
C
C   Convolve the pulse and the transfer function, and normalize.
C
            do l = 1, 256
               sum = 0.
               do j = 1, l
                  sum = sum + pwave(l - j + 1) * dble(spect(j))
               end do
               twave(l) = scale * sum * dt / (4. * pi * rho2)
            end do
C
C   Find the offset into the array and add the pulse to the time domain signal.
C
            i = (tdy - t) / dt
            do l = 1, 256
               if (l + i .ge. 1) then
                  wave(l + i) = wave(l + i) + twave(l)
               end if
            end do
         end do
      end do
      return
 2000 format("Convergence failed.  Limit = ",g14.6,", value = ",g14.6)

      end
```

H.2.1.8 Complex Function zk

```
C
C   Procedure name:   zk
C
C   Author:           J.C. Biard
C
C   Discussion:       This procedure calculates a complex wavenumber
C      from a real seed wavenumber, k, using Equation (60) in Volume I.
C      This complex wavenumber is used to introduce dispersion and
C      attenuation into the equations in which the results are used.  The
C      complex wavenumber can be determined for either of two different
C      media, as selected by the index n.
C         The real part of the complex wavenumber is considered to be
C      the dispersive wavenumber, for which the velocity of propagation
C      rises with the natural log of the frequency.  The imaginary part
C      is considered to be the attenuation, and rises with the exponential
C      of the frequency.
C         The calculations of the complex wavenumber make use of a
C      threshold radial frequency, wi, and a quality factor,
C      qi, (i = 1, 2).  Below the threshold frequency, there is no
C      dispersion or attenuation.  The quality factor is a unitless
C      quantity that describes the magnitude of the dispersion and
C      attenuation in a given medium.
C
C   Procedures called from this Module:
C
C      none
C
C   Formal arguments:
C
C      Input:
C
C         n         The medium for which the complex wavenumber is to be
C                      calculated.
C         k         The real wavenumber, which is used as a seed for the
C                      complex value.
C
C      Output:
C
C         zk        The complex wavenumber.
C
C   External variables:
C
C         nrefl     Number of reflection waves to calculate. (0<=nrefl<=6)
C         pi        Pi.
C         h         Thickness of the surface layer. (m)
C         c1        SH-wave velocity in the surface layer. (m / s)
C         rho1      Density of the surface layer. (kg / m)
C         c2        SH-wave velocity in the lower region. (m / s)
C         rho2      Density in the lower region. (kg / m)
C         a         Radius of the cavity. (m)
```

```
C        alpha     Shape factor for the incident pulse.
C        w0        Peak radial frequency for the incident pulse.
C        df        Frequency step. (Hz)
C        xc        X position of the cavity center.
C        yc        Y position of the cavity center.
C        q1        Quality factor for the surface layer.
C        w1        Cutoff radial frequency for the surface layer.
C        q2        Quality factor for the lower region.
C        w2        Cutoff radial frequency for the lower region.
C

      complex*16 function zk(n, k)

      implicit none

      integer*4  n
      real*8     k

      integer*4 nrefl
      real*8    pi, h, c1, rho1, c2, rho2, a, alpha, w0, df, xc, yc, q1,
     1          w1, q2, w2

      common /constants/ nrefl, pi, h, c1, rho1, c2, rho2, a,
     1                   alpha, w0, df, xc, yc, q1, w1, q2, w2

      real*8     beta, kp, wp

C
C  Calculate the complex wavenumber for medium 1.
C
      if (n .eq. 1) then
         wp = k * c1 / w1
C
C  If the wavenumber is over threshold, calculate the new velocity
C    and attenuation.
C
         if (wp .ge. 1.) then
            kp = k * (1. - log(wp) / pi / q1)
            beta = k * (1. - exp(-wp)) / 2. / q1
         else
            kp = k
            beta = 0.
         end if
      else
C
C  Calculate the complex wavenumber for medium 2.
C
         wp = k * c2 / w2
C
C  If the wavenumber is over threshold, calculate the new velocity
C    and attenuation.
C
         if (wp .ge. 1.) then
            kp = k * (1. - log(wp) / pi / q2)
            beta = k * (1. - exp(-wp)) / 2. / q2
         else
            kp = k
            beta = 0.
         end if
      end if
      zk = dcmplx(kp, beta)
      return

      end
```

H.2.2 Data Files

A sample input file and the first thirty lines of the resulting output file from the program synseis are shown below.

H.2.2.1 Sample Input

Below is a sample input file for the program synseis. The first line in the file specifies the number of reflections that are allowed to be processed, up to a maximum of six. For a value of 0, the result will be that of an infinite space. For a value of 1, the result will be that of a half-space with boundary conditions specified by other parameters in the file. For a value of 2 or greater, the program will model a half-space with a surface layer.

The second line specifies the velocity and density of the surface layer.

The third line specifies the quality factor of the surface layer and the cutoff radial frequency of the lossy behavior of the surface layer. If the cutoff is set very high, $\sim10^{30}$, the surface layer will be lossless.

The fourth line specifies the velocity and density of the lower region of the half-space.

The fifth line specifies the quality factor of the lower region and the cutoff radial frequency of the lossy behavior of the lower region. If the cutoff is set very high, $\sim10^{30}$, the lower region will be lossless.

The sixth line specifies the peak frequency, in Hertz, and the damping rate of the pulse. A value of .9 is strongly damped, and smaller values will cause the pulse to damp more slowly.

The seventh line specifies the step in frequency to use in all the calculations of the seismic waves in the frequency domain. This value is in units of Hertz.

The eighth line specifies the radius of the cavity in meters and the thickness of the surface layer in meters.

The ninth line specifies the location of the cavity in a coordinate system in which the half-space surface is the surface at y = 0 meters and in which the cavity and all sources and detectors must be in the region of negative y. The cavity location is given as an x, y pair, and it must not be located in the surface layer, which is considered to lie in the region from y = 0 meters to y = -h meters.

The tenth line specifies the source location in the coordinate system mentioned above. Like the cavity, the source must be located in the lower region.

The eleventh line specifies the starting location for the array of detectors at which the SH wave signal is calculated. This array is considered to run parallel to the x axis. The array must be located in the lower region or on the surface of the half-space, at y = 0 meters.

The last line specifies the numbei of detectors present in the detector array and the spacing between the detectors, in meters.

The example below is an exact copy of the input file used with the program synseis to generate the output in the next section. The comments on each line are not required, but may be there without disturbing the reading of the parameters by the program. The numerical values must be presented in the order and on the lines as shown below.

```
3                   $ No. of reflections. (0=space, 1=1/2space, >=2=layered)
500., 1500.         $ Layer 1: SH-wave velocity(m/s), density(kg/m**3).
50., .05            $          Q factor, reference radial frequency.
2500., 2500.        $ Layer 2: SH-wave velocity(m/s), density(kg/m**3).
100., .05           $          Q factor, reference radial frequency.
1000., .9           $ Pulse peak freq., damping factor.
39.78873577d0       $ Frequency step.
1., 5.              $ Cavity radius, layer depth (h >= 0.).
0., -50.            $ Cavity location (x, y).
-52., -6.           $ Source location (x, y).
1., -6.             $ Detector array start point (x,y). (y>-h or y=0)
1, 1.               $ No. of detectors, spacing.
```

H.2.2.2 Sample Output

The output below is an exact copy of the output generated by the program synseis when the input file is the one shown above. The listing below represents the first thirty lines of the output.

```
 1 4096 3 5.0 500.0 1500.0 2500.0 2500.0 6283.185307179585800
 .900000000000000 39.788735770000002 9.817477043201872E-005 1.0
 0.00000000000000E+000 -50.0 -52.0 -6.0 1.0 -6.0 50.0 5.00000000000000E-002
 100.0 5.00000000000000E-002 1.0
 1.0 1.83343267201207E-002 4.09793842635796E-002
 0.00000000000000E+000 -9.86040601716357E-006 -3.73468405469897E-005
 -7.41127820050589E-005 -1.05098616880071E-004 -1.14173362353061E-004
 -9.10977460277003E-005 -3.89472225515549E-005 1.89022072857182E-005
 5.13377893894403E-005 4.28637405384011E-005 1.57501389501543E-005
 5.05185341901528E-006 1.72098051512163E-005 -5.60444452183841E-006
 -8.91955288727754E-005 -1.34353715526207E-004 2.09600246275653E-004
```

```
3.16942661672617E-004 -6.72339541141405E-004 6.28431468436239E-003
3.50344989238494E-002 7.45644131768735E-002 8.94899625454235E-002
5.74394784148735E-002 -1.10606362129191E-002 -8.32881535613962E-002
-.125481864557382 -.120064508309060 -7.19679496071384E-002
-3.59457213114178E-003 5.70780148819543E-002 8.87964205581875E-002
8.44691452417778E-002 5.15940660037613E-002 6.85730712687943E-003
-3.18017360223787E-002 -5.20535645804432E-002 -5.05630857276737E-002
-3.23559256775042E-002 -7.26885042721200E-003 1.45741310113301E-002
2.63994691891793E-002 2.64215269418879E-002 1.73436876261574E-002
4.38783468713455E-003 -7.10772120059540E-003 -1.35689775237856E-002
-1.40038764466985E-002 -9.72365993731303E-003 -3.33190856932846E-003
2.48623312293225E-003 5.90465102652766E-003 6.36842672057483E-003
4.46687408454538E-003 1.44550730598057E-003 -1.38489499581388E-003
-3.11258612595481E-003 -3.43422291666952E-003 -2.60189768768065E-003
-1.19366238736904E-003 1.67627908958156E-004 1.03727376430705E-003
1.25286085709700E-003 9.15473387332386E-004 2.86564641272363E-004
-3.42686667695489E-004 -7.58024277573733E-004 -8.74601036794633E-004
-7.31383572196355E-004 -4.44758223389613E-004 -1.48244336254333E-004
5.72093419695609E-005 1.28399622417473E-004 7.88181535909211E-005
-4.10682906717466E-005 -1.70674346539787E-004 -2.65149577682537E-004
-3.04561214208585E-004 -2.90648806702587E-004 -2.40871529864743E-004
-1.78922213672084E-004 -1.25704399895347E-004 -9.37985761310485E-005
-8.63435174374761E-005 -9.88955480004431E-005 -1.22283473354508E-004
-1.45443228457183E-004 -1.58950926514508E-004 -1.59127788009629E-004
-1.50338460309153E-004 -1.40457626177698E-004 -1.29568410935122E-004
-1.05941034393917E-004 -9.35144899234639E-005 -7.45475467721954E-005
4.71950815358815E-004 2.54228770677183E-003 6.22662776085143E-003
9.66098601708087E-003 1.01414874721488E-002 6.27773283276044E-003
-9.05278139870855E-004 -8.53631966598725E-003 -1.34643777973222E-002
-1.37967868755026E-002 -9.65463042458288E-003 -2.92443221398139E-003
3.71768489576844E-003 7.96889631308235E-003 8.73284508063948E-003
6.34246611993576E-003 2.19481784404365E-003 -1.94947863538670E-003
-4.66935413178174E-003 -5.32398772451678E-003 -4.12148366907896E-003
-1.86842469331284E-003 4.40570625435726E-004 2.01082837235533E-003
2.47945519089527E-003 1.94361050229078E-003 8.21435863901723E-004
-3.66795321252789E-004 -1.20178960803560E-003 -1.48518179448681E-003
-1.25434602070362E-003 -7.12334403280661E-004 -1.17936535613221E-004
3.16357458824037E-004 4.85282246333699E-004 3.99868929222058E-004
1.54475246850195E-004 -1.26372315875142E-004 -3.37990752963346E-004
-4.26034762024539E-004 -3.91997569976660E-004 -2.78266548616755E-004
-1.43144880069879E-004 -3.69718114397421E-005 1.28243673593586E-005
5.00642189898001E-006 -4.19214407176348E-005 -1.01165464698855E-004
-1.48899425077008E-004 -1.71543058813405E-004 -1.67625900343532E-004
-1.45041265844154E-004 -1.15866230010875E-004 -9.11668470429718E-005
-7.75455582583915E-005 -7.61175122821318E-005 -8.36050544654795E-005
-9.46222420488954E-005 -1.04064948225721E-004 -1.08764734324236E-004
-1.08031982523774E-004 -1.03204260996270E-004 -9.66343329190498E-005
-9.06241584103258E-005 -8.66730962617657E-005 -8.51948321790042E-005
-8.56656130558596E-005 -8.70494844254843E-005 -8.82920652748281E-005
```

H.3 Graphics Programs

This section contains the programs and subroutines for producing graphics plots of the data computed by the modeling programs. They are written in the C language to allow for a greater flexibility in manipulating the data and to take advantage of the same language interface allowed in the use of the starbase graphics package subroutines, which are also written in C.

The main program segments for each different program are included under separate sections, while subroutines common to all the programs are included in another section.

H.3.1 Graphics for Plane Wave Results

For the plane wave modeling programs, the data generated by each program was different enough that a separate graphics program was written for each modeling program. Subroutines common to all of these programs are included in Section H.3.3.

H.3.1.1 graph_csect

```
/*
   Procedure          : graph_csect

   Author             : J.C. Biard

   Discussion         :
         This procedure draws plots of the data from the program csect.

   Invocation         :
         The procedure is run by typing

              graph_csect ifile gdev gdriver

      where ifile is the data file output by csect, gdev is the
      name of the graphics device special file, and gdriver
      is the driver name for the graphics device.

   External variables : none

   Procedures called from this module :
      axis               Draws axes and tic marks for the plot.
      character_size     Sets the size of the characters.
      init_coords        Initializes the plotting surface.
      x_labels           Draws x axis labels and titles the plot.
      y_labels           Draws y axis labels to the left of the axis.
      y_labels_r         Draws y axis labels to the right of the axis.
*/

#include <stdio.h>
#include <starbase.c.h>
```

```
main(argc, argv)
int argc;
char *argv[];
{
   char *xtitle, *ytitle, *title;
   int i, n, gfd;
   float x1, x2, y1, y2, dum;
   float wl, wr, wb, wt, xstep, ystep;
   float line[250], *aline, extent[12];
   double ka[125], sigt[125]:;
   FILE *fp:;

   FILE *fopen();

/*
   Open the input data file.
*/
   fp = fopen(argv[1], "r");
/*
   Read the data from the file into the appropriate arrays until the end
      of the file.
*/
   n = 0;
   while (fscanf(fp, "%lg%lg%*lg%*lg", &ka[n], &sigt[n]) > 0)
   {
      n++;
   }
/*
   Close the input data file.
*/
   fclose(fp);

/*
   Open the graphics device special file and draw the axes and labels on
      the graphics surface.
*/
   gfd = gopen (argv[2], OUTDEV, argv[3], RESET_DEVICE|INIT);
   wl = 0.;
   wr = 10.;
   wb = 0.;
   wt = 2.;
   xstep = 1.;
   ystep = .5;
   xtitle = "ka";
   ytitle = "q";
   title = "SH-WAVE SCATTERING CROSS-SECTION";
   init_coords(gfd, wl, wb, wr, wt, .15, .15, .05, .05);
   wc_to_vdc(gfd, wl, wb, 0., &x1, &y1, &dum);
   wc_to_vdc(gfd, wr, wt, 0., &x2, &y2, &dum);
   clip_rectangle(gfd, x1, x2, y1, y2);
   text_color_index(gfd, 5);
   line_color_index(gfd, 5);
   axis(gfd, 0., 0., xstep, ystep);
   line_color_index(gfd, 6);
```

```
    move2d(gfd, 2., wb);
    draw2d(gfd, 2., wt);
    move2d(gfd, 4., wb);
    draw2d(gfd, 4., wt);
    move2d(gfd, 6., 1.);
    draw2d(gfd, 6., wt);
    move2d(gfd, 8., 1.);
    draw2d(gfd, 8., wt);
    move2d(gfd, 10., wb);
    draw2d(gfd, 10., wt);
    move2d(gfd, wl, .5);
    draw2d(gfd, 4., .5);
    move2d(gfd, wl, 1.);
    draw2d(gfd, wr, 1.);
    move2d(gfd, wl, 1.5);
    draw2d(gfd, wr, 1.5);
    draw2d(gfd, wl, 2.);
    draw2d(gfd, wr, 2.);
    line_color_index(gfd, 5);
    character_size(gfd, .05, .5);
    inquire_text_extent(gfd, "A", WORLD_COORDINATE_TEXT, extent);
    x1 - extent[9] - extent[3];
    y1 = extent[10] - extent[4];
    text_alignment(gfd, TA_LEFT, TA_BOTTOM, 0., 0.);
    x2 = 4. + x1;
    y2 = 1. - 3. * y1;
    test2d(gfd, x2, y2, "TOTAL REFLECTED ENERGY", WORLD_COORDINATE_TEXT, FALSE);
    y2 -= y1;
    text2d(gfd, x2, y2, "PLANE WAVE SOURCE", WORLD_COORDINATE_TEST, FALSE);
    y2 -= y1;
    character_size(gfd, .06, .5);
    text2d(gfd, x2, y2, "a", WORLD_COORDINATE_TEXT, FALSE);
    character_size(gfd, .05, .5);
    x2 += x1;
    text2d(gfd, x2, y2, " = 1 METER", WORLD_COORDINATE_TEXT, FALSE);
    x2 -= x1;
    y2 -= y1;
    text2d(gfd, x2, y2, "V", WORLD_COORDINATE_TEXT, FALSE);
    x2 += x1;
    text_alignment(gfd, TA_LEFT, TA_HALF, 0., 0.);
    text2d(gfd, x2, y2, "s", WORLD_COORDINATE_TEXT, FALSE);
    x2 += x1;
    text_alignment(gfd, TA_LEFT, TA_BOTTOM, 0., 0.);
    text2d(gfd, x2, y2, " = 2,500 m/s", WORLD_COORDINATE_TEXT, FALSE);
    x_labels(gfd, 0., 0., xstep, 0., xstep, wl, wr, "%4.1f", xtitle, title);
    y_labels(gfd, 0., 0., ystep, 0., ystep, wb, wt, "%3.1f", ytitle);
/*
    Draw the cross-section curve.
*/
    aline = &line[0];
    for (i = 0 ; i <= n ; i++)
```

```
   {
      *aline++ = ka[i];
      *aline++ = sigt[i];
   }
   line_color_index(gfd, 2);
   polyline2d(gfd, line, n, FALSE);
/*
   Close the graphics device.
*/
   gclose(gfd);
}
```

H.3.1.2 graph_dispa

```
/*
   Procedure          : graph_dispa

   Author             : J.C. Biard

   Discussion         :
         This procedure draws plots of the data from the program dispa.

   Invocation         :
         The procedure is run by typing

            graph_dispa ifile gdev gdriver

      where ifile is the data file output by dispa, gdev is the
      name of the graphics device special file, and gdriver
      is the driver name for the graphics device.

   External Variables : none

   Procedures called from this module :
      axis                Draws axes and tic marks for the plot.
      character_size      Sets the size of the characters.
      init_coords         Initializes the plotting surface.
      x_labels            Draws x axis labels and titles the plot.
      y_labels            Draws y axis labels to the left of the axis.
      y_labels_r          Draws y axis labels to the right of the axis.
*/

#include <stdio.h>
#include <starbase.c.h>

main(argc, argv)
int argc;
char *argv[];
{
   char *xtitle, *ytitle, title[132], buffer[132], *title2, buffer2[80];
   int i. n, gfd, virgin;
   float x1, x2, y1, y2, dum, pi, extent[12];
   float wl, wr, wb, wt, xstep, ystep;
   float line[500], *aline;
   double ka[250], amp[250], phase[250];
   FILE *fp;

   char *strcpy();
   FILE *fopen();

   pi = 3.141592654;
   title2 = "SH-WAVE SCATTERING VS. FREQUENCY";
   virgin = 0;
/*
   Open the graphics device special file and select the default pens.
*/
   gfd = gopen(argv[2], OUTDEV, argv[3], RESET_DEVICE|INIT);
   text_color_index(gfd, 5);
   line_color_index(gfd, 5);
/*
   Open the data input file.
*/
   fp = fopen(argv[1], "r");
/*
   Read one line of data at a time until the end of the file.
*/
   while (fscanf(fp, "%[A-Za-z0-9,. +-=]\n", buffer) > 0)
   {
/*
   If the line is a title line, get the title or plot the accumulated data.
*/
      if (buffer[0] == 'S')
      {
/*
   If this is the first title line, get the title only.
*/
         if (virgin == 0)
         {
            fscanf(fp, "\n%[A-Za-z0-9,. +-=]\n", title);
            strcpy(title, buffer);
            virgin = 1;
         }
/*
   If this is not the first title line, plot the data.
*/
         else
         {
/*
   Get a new graphics surface and draw the axes and labels for the amplitude
      part of the plot.
*/
            clear_view_surface(gfd);
            wl = 0.;
            wr = 10.;
            wb = 0.;
            wt = 2.2001;
      xstep = 1.;
      ystep = .2;
      xtitle = "ka";
      ytitle = "AMPLITUDE";
      init_coords(gfd, wl, wb, wr, wt, .15, .15, .18, .05);
      wc_to_vdc(gfd, wl, wb, 0., &x1, &y1, &dum);
      wc_to_vdc(gfd, wr, wt, 0., &x2, &y2, &dum);
      clip_rectangle(gfd, x1, x2, y1, y2);
      axis(gfd, 0., 0., xstep, ystep);
      x_labels(gfd, 0., 0., xstep, 0., xstep, wl, wr, "%2.0f.",
          xtitle, title2);
      y_labels(gfd, 0., 0., ystep, 0., ystep, wb, wt, "%3.1f", ytitle);
      text_alignment(gfd, TA_LEFT, TA_BOTTOM, 0., 0.);
      character_size(gfd, .05, .5);
      sscanf(&title[40], "%f", &x1);
      sprintf(buffer2, "PHI = %2.0f. DEG.", x1 * 180. / pi);
      inquire_text_extent(gfd, "A", WORLD_COORDINATE_TEXT, extent);
      x1 = extent[9] - extent[3];
      y1 = extent[10] - extent[4];
      x2 = 0. + 2. * x1;
      y2 = .45;
      text2d(gfd, x2, y2, buffer2, WORLD_COORDINATE_TEXT, FALSE);
      y2 -= y1;
      character_size(gfd, .06, .5);
      text2d(gfd, x2, y2, "a", WORLD_COORDINATE_TEXT, FALSE);
      character_size(gfd, .05, .5);
      x2 += x1;
      text2d(gfd, x2, y2, "   = 1 METER", WORLD_COORDINATE_TEXT, FALSE);
      x2 -= x1;
      y2 -= y1;
      text2d(gfd, x2, y2, "V", WORLD_COORDINATE_TEXT, FALSE);
      x2 += x1;
      text_alignment(gfd, TA_LEFT, TA_HALF, 0., 0.);
      text2d(gfd, x2, y2, "s", WORLD_COORDINATE_TEXT, FALSE);
      x2 += x1;
      text_alignment(gfd, TA_LEFT, TA_BOTTOM, 0., 0.);
      text2d(gfd, x2, y2, "  = 2,500 m/s", WORLD_COORDINATE_TEXT, FALSE);
      y2 += y1;
      x2 = 8.;
      text_alignment(gfd, TA_RIGHT, TA_BOTTOM, 0., 0.);
      text2d(gfd, x2, y2, "AMP.  ", WORLD_COORDINATE_TEXT, FALSE);
      line_color_index(gfd, 2);
      move2d(gfd, x2, y2 + .5 * y1);
      draw2d(gfd, 9., y2 + .5 * y1);
      y2 -= y1;
      text2d(gfd, x2, y2, "PHASE ", WORLD_COORDINATE_TEXT, FALSE);
      line_color_index(gfd, 3);
      move2d(gfd, x2, y2 + .5 * y1);
      draw2d(gfd, 9., y2 + .5 * y1);
      line_color_index(gfd, 5);
/*
  Draw the amplitude data curve.
*/
```

```
        aline = &line[0];
        for (i = 0 ; i <= n ; i++)
        {
           *aline++ = ka[i];
           *aline++ = amp[i];
        }
        line_color_index(gfd, 2);
        polyline2d(gfd, line, n, FALSE);
/*
  Draw axes and labels for the phase part of the plot.
*/
        line_color_index(gfd, 5);
        wl = 0.;
        wr = 10.;
        wb = -180.;
        wt = 180.;
        xstep = 0.;
        ystep = 45.;
        xtitle = "ka";
        ytitle = "PHASE";
        init_coords(gfd, wl, wb, wr, wt, .15, .15, .18, .05);
        wc_to_vdc(gfd, wl, wb, 0., &xl, &yl, &dum);
        wc_to_vdc(gfd, wr, wt, 0., &x2, &y2, &dum);
        clip_rectangle(gfd, xl, x2, yl, y2);
        axis(gfd, wr - .00001, -180., xstep, ystep);
        y_labels_r(gfd, wr, 0., ystep, 0., ystep, wb, wt, "%5.0f.", ytitle);
/*
  Draw the phase data curve.
*/
        aline = &line[0];
        for (i = 0 ; i <= n ; i++)
        {
           *aline++;
           *aline++ = 180. * phase[i] / pi;
        }
        line_color_index(gfd, 3);
        polyline2d(gfd, line, n, FALSE);
        line_color_index(gfd, 5);
/*
  Read the next line of input.
*/
        fscanf(fp, "\n%[A-Za-z0-9,. +-=]\n", title);
        strcpy(title, buffer);
      }
      n = 0;
    }
/*
  Store the data from the current line into the appropriate arrays.
*/
    else
    {
      sscanf(buffer, "%*d %lg %lg %lg %*lg", &ka[n], &amp[n], &phase[n]);
      n++;
    }
  }
```

```
/*
   Get a new graphics surface and draw labels and axes for the last plot.
*/
   clear_view_surface(gfd);
   wl = 0.;
   wr = 10.;
   wb = 0.;
   wt = 2.2001;
   xstep = 1.;
   ystep = .2;
   xtitle = "ka";
   ytitle = "AMPLITUDE";
   init_coords(gfd, wl, wb, wr, wt, .15, .15, .18, .05);
   wc_to_vdc(gfd, wl, wb, 0., &xl, &yl, &dum);
   wc_to_vdc(gfd, wr, wt, 0., &x2, &y2, &dum);
   clip_rectangle(gfd, xl, x2, yl, y2);
   axis(gfd, 0., 0., xstep, ystep);
   x_labels(gfd, 0., 0., xstep, 0., xstep, wl, wr, "%2.0f.", xtitle, title2);
   y_labels(gfd, 0., 0., ystep, 0., ystep, wb, wt, "%3.1f", ytitle);
   text_alignment(gfd, TA_LEFT, TA_BOTTOM, 0., 0.);
   character_size(gfd, .05, .5);
   sscanf(&title[40], "%f", &xl);
   sprintf(buffer2, "PHI = %2.0f. DEG.", xl * 180. / pi);
   inquire_text_extent(gfd, "A", WORLD_COORDINATE_TEXT, extent);
   xl = extent[9] - extent[3];
   yl = extent[10] - extent[4];
   x2 = 0. + 2. * xl;
   y2 = .45;
   text2d(gfd, x2, y2, buffer2, WORLD_COORDINATE_TEXT, FALSE);
   y2 -= yl;
   character_size(gfd, .06, .5);
   text2d(gfd, x2, y2, "a", WORLD_COORDINATE_TEXT, FALSE);
   character_size(gfd, .05, .5);
   x2 += xl;
   text2d(gfd, x2, y2, "  = 1 METER", WORLD_COORDINATE_TEXT, FALSE);
   x2 -= xl;
   y2 -= yl;
   text2d(gfd, x2, y2, "V", WORLD_COORDINATE_TEXT, FALSE);
   x2 += xl;
   text_alignment(gfd, TA_LEFT, TA_HALF, 0., 0.);
   text2d(gfd, x2, y2, "s", WORLD_COORDINATE_TEXT, FALSE);
   x2 += xl;
   text_alignment(gfd, TA_LEFT, TA_BOTTOM, 0., 0.);
   text2d(gfd, x2, y2, "  = 2,500 m/s", WORLD_COORDINATE_TEXT, FALSE);
   y2 += yl;
   x2 = 8.;
   text_alignment(gfd, TA_RIGHT, TA_BOTTOM, 0., 0.);
   text2d(gfd, x2, y2, "AMP.  ", WORLD_COORDINATE_TEXT, FALSE);
   line_color_index(gfd, 2);
   move2d(gfd, x2, y2 + .5 * yl);
   draw2d(gfd, 9., y2 + .5 * yl);
   y2 -= yl;
   text2d(gfd, x2, y2, "PHASE ", WORLD_COORDINATE_TEXT, FALSE);
   line_color_index(gfd, 3);
   move2d(gfd, x2, y2 + .5 * y1);
   draw2d(gfd, 9., y2 + .5 * y1);
   line_color_index(gfd, 5);
/*
   Draw the amplitude data curve.
*/
   aline = &line[0];
   for (i = 0 ; i <= n ; i++)
   {
      *aline++ = ka[i];
      *aline++ = amp[i];
   }
   line_color_index(gfd, 2);
   polyline2d(gfd, line, n, FALSE);
/*
   Draw the axes and labels for the phase data.
*/
   line_color_index(gfd, 5);
   wl = 0.;
   wr = 10.;
   wb = -180.;
   wt = 180.;
   xstep = 0.;
   ystep = 45.;
   xtitle = "ka";
   ytitle = "PHASE";
   init_coords(gfd, wl, wb, wr, wt, .15, .15, .18, .05);
   wc_to_vdc(gfd, wl, wb, 0., &x1, &y1, &dum);
   wc_to_vdc(gfd, wr, wt, 0., &x2, &y2, &dum);
   clip_rectangle(gfd, x1, x2, y1, y2);
   axis(gfd, wr - .00001, -180., xstep, ystep);
   y_labels_r(gfd, wr, 0., ystep, 0., ystep, wb, wt, "%5.0f.", ytitle);
/*
   Draw the phase data curve.
*/
   aline = &line[0];
   for (i = 0 ; i <= n ; i++)
   {
      *aline++;
      *aline++ = 180. * phase[i] / pi;
   }
   line_color_index(gfd, 3);
   polyline2d(gfd, line, n, FALSE);
   line_color_index(gfd, 5);
/*
   Close the input file and the graphics device.
*/
   fclose(fp);
   gclose(gfd);
}
```

H.3.1.3 graph_dispb

```
/*
   Procedure          : graph_dispb

   Author             : J.C. Biard

   Discussion         :
         This procedure draws plots of the data from the program dispb.

   Invocation         :
         The procedure is run by typing

            graph_dispb ifile gdev gdriver

      where ifile is the data file output by dispb, gdev is the
      name of the graphics device special file, and gdriver
      is the driver name for the graphics device.

   External Variables : none

   Procedures called from this module :
      axis                Draws axes and tic marks for the plot.
      character_size      Sets the size of the characters.
      init_coords         Initializes the plotting surface.
      x_labels            Draws x axis labels and titles the plot.
      y_labels            Draws y axis labels to the left of the axis.
      y_labels_r          Draws y axis labels to the right of the axis.
*/

#include <stdio.h>
#include <starbase.c.h>

main(argc, argv)
int argc;
char *argv[];
{
   char *xtitle, *ytitle, title[a32], buffer[132], buffer2[132];
   int i, n, gfd, virgin, ncurve;
   float x1, x2, y1, y2, dum;
   float wl, wr, wb, wt, xstep, ystep;
   float line[500], *aline;
   double ra[250], amp[250], phase[250];
   FILE *fp;

   char *strcpy(), *strcat();
   FILE *fopen();

   virgin = 0;
/*
   Open the graphics device and the input file.
*/
   gfd = gopen(argv[2], OUTDEV, argv[3], RESET_DEVICE|INIT);
   fp = fopen(argv[1], "r");
   ncurve = 0;
```

```
/*
   Read a line from the input file until the end of the file.
*/
   while (fscanf(fp, "%[A-Za-z0-9,. +-=]\n", buffer) > 0)
   {
/*
   If a title line is read then either get the title or plot data.
*/
      if (buffer[0] == 'D')
      {
/*
   If this is the first title line, get the title.
*/
         if (virgin == 0)
         {
            fscanf(fp, "\n%[A-Za-z0-9,. +-=]\n", buffer2);
            sscanf(buffer, "%[^,]%*26c%26c", title, buffer2);
            sprintf(title, "%s, %26c", title, buffer2);
            virgin = 1;
         }
/*
   If this is not the first title line, plot the data.
*/
         else
         {
            ncurve++;
/*
   Start a new plot for every third curve.
*/
            if (ncurve % 3 == 1)
            {
/*
   Get a new page and draw labels and axes.
*/
               clear_view_surface(gfd);
               line_type(gfd, SOLID);
               wl = 0.;
               wr = 50.;
               wb = 0.;
               wt = 2.2;
               xstep = 5.;
               ystep = .2;
               xtitle = "r/a";
               ytitle = "Amplitude";
               init_coords(gfd, wl, wb, wr, wt, .15, .15, .05, .05);
               wc_to_vdc(gfd, wl, wb, 0., &x1, &y1, &dum);
               wc_to_vdc(gfd, wr, wt, 0., &x2, &y2, &dum);
               clip_rectangle(gfd, x1, x2, y1, y2);
               axis(gfd, 0., 0., xstep, ystep);
               x_labels(gfd, 0., 0., xstep, 0., xstep, wl, wr, "%2.0f.", xtitle,
                  title);
               y_labels(gfd, 0., 0., ystep, 0., ystep, wb, wt, "%3.1f", ytitle);
            }
/*
   Select the line style appropriate for each curve number in the plot.
*/
            switch (ncurve % 3)
            {
               case 0:
                  line_type(gfd, DASH_DOT_DOT);
                  break;
               case 1:
                  line_type(gfd, SOLID);
                  break;
               case 2:
                  line_type(gfd, DASH_DOT);
                  break;
            }
/*
   Draw the curve.
*/
            aline = &line[0];
            for (i = 0 ; i <= n ; i++)
            {
               *aline++ = ra[i];
               *aline++ = amp[i];
            }
            polyline2d(gfd, line, n, FALSE);
/*
   Read the next line.
*/
            fscanf(fp, "\n%[A-Za-z0-9,. +-=]\n", buffer2);
/*
   If the next set of data will start a new plot, get the title.
*/
            if (ncurve % 3 == 0)
            {
               sscanf(buffer, "%[^,]%*26c%26c", title, buffer2);
               sprintf(title, "%s, %26c", title, buffer2);
            }
         }
         n = 0;
      }
/*
   Store the input data in the appropriate locations.
*/
      else
      {
         sscanf(buffer, "%*d %lg %*lg %lg %*lg", &ra[n], &amp[n]);
         n++;
      }
   }
```

```
/*
   Draw the last line in the file.
*/
   line_type(gfd, DASH_DOT_DOT);
   aline = &line[0];
   for (i = 0 ; i <= n ; i++)
   {
      *aline++ = ra[i];
      *aline++ = amp[i];
   }
   polyline2d(gfd, line, n, FALSE);
/*
   Close the graphics device and the input file.
*/
   fclose(fp);
   gclose(gfd);
}
```

H.3.1.4 graph_disped

```
/*
   Procedure          : graph_disped

   Author             : J.C. Biard

   Discussion         :
         This procedure draws plots of the data from the program disped.

   Invocation         :
         The procedure is run by typing

            graph_disped ifile gdev gdriver

      where ifile is the data file output by disped, gdev is the
      name of the graphics device special file, and gdriver
      is the driver name for the graphics device.

   External Variables : none

   Procedures called from this module :
      axis                 Draws axes and tic marks for the plot.
      character_size       Sets the size of the characters.
      init_coords          Initializes the plotting surface.
      x_labels             Draws x axis labels and titles the plot.
      y_labels             Draws y axis labels to the left of the axis.
      y_labels_r           Draws y axis labels to the right of the axis.
*/
```

```
#include <math.h>
#include <stdio.h>
#include <starbase.c.h>

main(argc, argv)
int argc;
char *argv[];
{
   char *xtitle, *ytitle, title[132], buffer[132], buffer2[132], *title2;
   int i, n, gfd, virgin, ncurve;
   float x1, x2, y1, y2, dum, extent[12];
   float wl, wr, wb, wt, xstep, ystep, r, psi, pi;
   float line[500], *aline;
   double phi[250], amp[250], phase[250];
   FILE *fp;

   char *strcpy(), *strcat();
   double sin(), cos();
   FILE *fopen();

   pi = 3.141592654;
   virgin = 0;
/*
   Open the graphics device.
*/
   gfd = gopen(argv[2], OUTDEV, argv[3], RESET_DEVICE|INIT);
/*
   Set the pens to use for lines and text.
*/
   text_color_index(gfd, 5);
   line_color_index(gfd, 5);
/*
   Open the input file.
*/
   fp = fopen(argv[1], "r");
   ncurve = 0;
/*
   Read an input file line until the end of the file.
*/
   while (fscanf(fp, "%[A-Za-z0-9,. +-=\f]\n", buffer) > 0)
   {
/*
   If a title line is seen get the title or plot the accumulated data.
*/
      if (buffer[0] == '\f' || buffer[0] == 'D')
      {
/*
   If this is the first title line, then get the title.
*/
         if (virgin == 0)
         {
            fscanf(fp, "\n%[A-Za-z0-9,. +-=\f]\n", buffer2);
            strcpy(title, buffer);
            virgin = 1;
         }
```

```
/*
   If this is not the first title line, then plot the data.
*/
         else
         {
/*
   Plot three sets of data on a page.
*/
            ncurve++;
            if (ncurve % 3 == 1)
            {
/*
   If this is to start a new plot, then get a new surface and draw axes
      and labels.
*/
               clear_view_surface(gfd);
               wl = -2.;
               wr = 2.;
               wb = 0.;
               wt = 2.;
               xstep = .5;
               ystep = 0.;
               xtitle = "AMPLITUDE (REL.)";
               ytitle = " ";
               title2 = "TOTAL SH-WAVE POLAR AMPLITUDE SCATTERING";
               init_coords(gfd, wl, wb, wr, wt, .15, .15, .15, .05);
               wc_to_vdc(gfd, wl, wb, 0., &x1, &y1, &dum);
               wc_to_vdc(gfd, wr, wt, 0., &x2, &y2, &dum);
               clip_rectangle(gfd, x1, x2, y1, y2);
               axis(gfd, -3., 0., xstep, ystep);
               x_labels(gfd, 0., 0., xstep, 0., xstep, wl, wr, "%3.1f",
                    xtitle, title2);
               clip_indicator(gfd, CLIP_TO_VDC);
/*
   Draw curved dotted line of radius .2.
*/
               line_color_index(gfd, 6);
               r = .2;
               move2d(gfd, r, 0.);
               for (psi = 0. ; psi <= 2. * pi + .001 ; psi += pi / 90.)
               {
                  x1 = r * cos(psi);
                  y1 = r * sin(psi);
                  draw2d(gfd, x1, y1);
               }
/*
   Draw curved dotted lines of radius 1. and 2.
*/
               for (r = 1. ; r <= 2. ; r++)
               {
                  move2d(gfd, r, 0.);
                  for (psi = 0. ; psi <= pi + .001 ; psi += pi / 300.)
```

```
        axis(gfd, 0., 0., xstep, ystep);
        line_color_index(gfd, 6);
```

```
                  {
                     x1 = r * cos(psi);
                     y1 = r * sin(psi);
                     draw2d(gfd, x1, y1);
                  }
               }
/*
   Draw radial lines and angle labels at every 15 degrees around the
   radius 2. circle.
*/
               r = 2.;
               character_size(gfd, .035, .5);
               inquire_text_extent(gfd, "A", WORLD_COORDINATE_TEXT, extent);
               x2 = extent[9] - extent[3];
               for (psi = 0. ; psi <= pi + .0001 ; psi += pi / 6.)
               {
                  x1 = r * cos(psi);
                  y1 = r * sin(psi);
                  move2d(gfd, 0., 0.);
                  draw2d(gfd, x1, y1);
                  sprintf(buffer2, "%1.0f", psi * 180. / pi);
                  if (psi < pi / 2. - .1)
                  {
                     text_alignment(gfd, TA_LEFT, TA_BOTTOM, 0., 0.);
                     x1 += .5 * x2;
                     text2d(gfd, x1, y1, buffer2, WORLD_COORDINATE_TEXT, FALSE);
                  }
                  else if (psi - pi / 2. < .01)
                  {
                     text_alignment(gfd, TA_CENTER, TA_BOTTOM, 0., 0.);
                     text2d(gfd, x1, y1, buffer2, WORLD_COORDINATE_TEXT, FALSE);
                  }
                  else
                  {
                     text_alignment(gfd, TA_RIGHT, TA_BOTTOM, 0., 0.);
                     x1 -= .5 * x2;
                     text2d(gfd, x1, y1, buffer2, WORLD_COORDINATE_TEXT, FALSE);
                  }
               }
/*
   Draw graph legends.
*/
               line_color_index(gfd, 5);
               text_alignment(gfd, TA_LEFT, TA_BOTTOM, 0., 0.);
               character_size(gfd, .05, .5);
               sscanf(&title[47], "%f", &x1);
               sprintf(buffer2, " = %1.0f.", x1);
               inquire_text_extent(gfd, "A", WORLD_COORDINATE_TEXT, extent);
               x1 = extent[9] - extent[3];
               y1 = extent[10] - extent[4];
               x2 = wl;
               y2 = wt + 4. * y1;
               character_size(gfd, .06, .5);
               text2d(gfd, x2, y2, "r/a", WORLD_COORDINATE_TEXT, FALSE);
               character_size(gfd, .05, .5);
               x2 += 3. * xl;
               text2d(gfd, x2, y2, buffer2, WORLD_COORDINATE_TEXT, FALSE);
               x2 -= 3. * xl;
               y2 -= yl;
               character_size(gfd, .06, .5);
               text2d(gfd, x2, y2, "a", WORLD_COORDINATE_TEXT, FALSE);
               character_size(gfd, .05, .5);
               x2 += xl;
               text2d(gfd, x2, y2, "   = 1 METER", WORLD_COORDINATE_TEXT, FALSE);
               x2 -= xl;
               y2 -= yl;
               text2d(gfd, x2, y2, "V", WORLD_COORDINATE_TEXT, FALSE);
               x2 += xl;
               text_alignment(gfd, TA_LEFT, TA_HALF, 0., 0.);
               text2d(gfd, x2, y2, "s", WORLD_COORDINATE_TEXT, FALSE);
               x2 += xl;
               text_alignment(gfd, TA_LEFT, TA_BOTTOM, 0., 0.);
               text2d(gfd, x2, y2, "  = 2,500 m/s", WORLD_COORDINATE_TEXT, FALSE);
               y2 = wt + 4. * yl;
               x2 = 1.5;
               text_alignment(gfd, TA_RIGHT, TA_BOTTOM, 0., 0.);
               text2d(gfd, x2, y2, "ka =  .5", WORLD_COORDINATE_TEXT, FALSE);
               line_color_index(gfd, 2);
               move2d(gfd, x2 + .5 * xl, y2 + .5 * yl);
               draw2d(gfd, 2., y2 + .5 * yl);
               y2 -= yl;
               text2d(gfd, x2, y2, "ka = 1.0", WORLD_COORDINATE_TEXT, FALSE);
               line_color_index(gfd, 3);
               move2d(gfd, x2 + .5 * xl, y2 + .5 * yl);
               draw2d(gfd, 2., y2 + .5 * yl);
               y2 -= yl;
               text2d(gfd, x2, y2, "ka = 2.0", WORLD_COORDINATE_TEXT, FALSE);
               line_color_index(gfd, 4);
               move2d(gfd, x2 + .5 * xl, y2 + .5 * yl);
               draw2d(gfd, 2., y2 + .5 * yl);
               clip_indicator(gfd, CLIP_TO_RECT);
               line_color_index(gfd, 5);
            }
/*
   Select the proper pen color for the current curve to plot.
*/
            switch (ncurve % 3)
            {
               case 0:
                  line_color_index(gfd, 4);
                  break;
               case 1:
                  line_color_index(gfd, 2);
                  break;
               case 2:
                  line_color_index(gfd, 3);
                  break;
            }
```

```
/*
   Draw the curve.
*/
            aline = &line[0];
            for (i = 0 ; i <= n ; i++)
            {
               *aline++ = amp[i] * cos(phi[i]);
               *aline++ = amp[i] * sin(phi[i]);
            }
            polyline2d(gfd, line, n, FALSE);
            line_color_index(gfd, 5);
/*
   Read the next input line.
*/
            fscanf(fp, "\n%[A-Za-z0-9,. +-=\f]\n", buffer2);
/*
   If the next curve will start a new graph, save the title line.
*/
            if (ncurve % 3 == 0)
            {
               strcpy(title, buffer);
            }
         }
         n = 0;
      }
/*
   Store the input data in the proper arrays.
*/
      else
      {
         sscanf(buffer, "%*d %lg %lg %*lg", &phi[n], &amp[n]);
         n++;
      }
   }
/*
   Plot the last curve in the file.
*/
   line_color_index(gfd, 4);
   aline = &line[0];
   for (i = 0 ; i <= n ; i++)
   {
      *aline++ = amp[i] * cos(phi[i]);
      *aline++ = amp[i] * sin(phi[i]);
   }
   polyline2d(gfd, line, n, FALSE);
   clear_view_surface(gfd);
/*
   Close the input file and the graphics device.
*/
   fclose(fp);
   gclose(gfd);
}
```

H.3.1.5 graph_disp

```
/*
   Procedure          : graph_dispe

   Author             : J.C. Biard

   Discussion         :
         This procedure draws plots of the data from the program dispe.

   Invocation         :
         The procedure is run by typing

            graph_dispe ifile gdev gdriver

      where ifile is the data file output by dispe, gdev is the
      name of the graphics device special file, and gdriver
      is the driver name for the graphics device.

   External Variables : none

   Procedures called from this module :
      axis                Draws axes and tic marks for the plot.
      character_size      Sets the size of the characters.
      init_coords         Initializes the plotting surface.
      x_labels            Draws x axis labels and titles the plot.
      y_labels            Draws y axis labels to the left of the axis.
      y_labels_r          Draws y axis labels to the right of the axis.
*/

#include <stdio.h>
#include <starbase.c.h>

main(argc, argv)
int argc;
char *argv[];
{
   char *xtitle, *ytitle, title[132], buffer[132];
   int i, n, gfd, virgin;
   float x1, x2, y1, y2, dum;
   float wl, wr, wb, wt, xstep, ystep;
   float line[500], *aline;
   double x[250], amp[250], phase[250];
   FILE *fp;

   char *strcpy();
   FILE *fopen();

   virgin = 0;
```

```
/*
   Open the graphics device.
*/
   gfd = gopen(argv[2], OUTDEV, argv[3], RESET_DEVICE|INIT);
/*
   Open the input file for reading.
*/
   fp = fopen(argv[1], "r");
/*
   Read the file until the end and load the appropriate values.
*/
   while (fscanf(fp, "%[A-Za-z0-9,. +-=]\n", buffer) > 0)
   {
/*
   If a title line is found, then get the title or plot the data.
*/
      if (buffer[0] == 'D')
      {
/*
   If this is the first pass, get the title.
*/
         if (virgin == 0)
         {
            fscanf(fp, "\n%[A-Za-z0-9,. +-=]\n", title);
            strcpy(title, buffer);
            virgin = 1;
         }
/*
   If this is not the first pass, prepare to plot the data.
*/
         else
         {
/*
   Get a new screen and draw the axes and labels.
*/
            clear_view_surface(gfd);
            wl = -50.;
            wr = 50.;
            wb = 0.;
            wt = 2.2;
            xstep = 10.;
            ystep = .2;
            xtitle = "x (meters)";
            ytitle = "Amplitude";
            init_coords(gfd, wl, wb, wr, wt, .15, .15, .05, .05);
            wc_to_vdc(gfd, wl, wb, 0., &xl, &yl, &dum);
            wc_to_vdc(gfd, wr, wt, 0., &x2, &y2, &dum);
            clip_rectangle(gfd, xl, x2, yl, y2);
            axis(gfd, wl, 0., xstep, ystep);
            x_labels(gfd, wl, 0., xstep, wl, xstep, wl, wr, "%2.0f.", xtitle,
                 title);
            y_labels(gfd, wl, 0., ystep, 0., ystep, wb, wt, "%3.1f", ytitle);
/*
   Draw the amplitude data accumulated.
*/
            aline = &line[0];
            for (i = 0 ; i <= n ; i++)
            {
               *aline++ = x[i];
               *aline++ = amp[i];
            }
            polyline2d(gfd, line, n, FALSE);
/*
   Get a new screen and draw the axes and labels.
*/
            clear_view_surface(gfd);
            wl = -50.;
            wr = 50.;
            wb = -10.;
            wt = 10.;
            xstep = 10.;
            ystep = 2.;
            ytitle = "phase (radians)";
            init_coords(gfd, wl, wb, wr, wt, .15, .15, .05, .05);
            wc_to_vdc(gfd, wl, wb, 0., &xl, &yl, &dum);
            wc_to_vdc(gfd, wr, wt, 0., &x2, &y2, &dum);
            clip_rectangle(gfd, xl, x2, yl-.001, y2);
            axis(gfd, wl, wb, xstep, ystep);
            x_labels(gfd, wl, wb, xstep, wl, xstep, wl, wr, "%2.0f.", xtitle,
                 title);
            y_labels(gfd, wl, 0., ystep, 0., ystep, wb, wt, "%4.0f.", ytitle);
/*
   Draw the phase data accumulated.
*/
            aline = &line[0];
            for (i = 0 ; i <= n ; i++)
            {
               *aline++;
               *aline++ = phase[i];
            }
            polyline2d(gfd, line, n, FALSE);
            fscanf(fp, "\n%[A-Za-z0-9,. +-=]\n", title);
            strcpy(title, buffer);
         }
         n = 0;
      }
/*
   If there is not a title, then read the data.
*/
      else
      {
         sscanf(buffer, "%*d %lg %lg %lg %*lg", &x[n], &amp[n], &phase[n]);
         n++;
      }
   }
```

```
/*
   Get a new screen and draw the axes and labels.
*/
   clear_view_surface(gfd);
   wl = -50.;
   wr = 50.;
   wb = 0.;
   wt = 2.2;
   xstep = 10.;
   ystep = .2;
   ytitle = "Amplitude";
   init_coords(gfd, wl, wb, wr, wt, .15, .15, .05, .05);
   wc_to_vdc(gfd, wl, wb, 0., &xl, &yl, &dum);
   wc_to_vdc(gfd, wr, wt, 0., &x2, &y2, &dum);
   clip_rectangle(gfd, xl, x2, yl, y2);
   axis(gfd, wl, 0., xstep, ystep);
   x_labels(gfd, wl, 0., xstep, wl, xstep, wl, wr, "%2.0f.", xtitle, title);
   y_labels(gfd, wl, 0., ystep, 0., ystep, wb, wt, "%3.1f", ytitle);
/*
   Draw the amplitude data accumulated.
*/
   aline = &line[0];
   for (i = 0 ; i <= n ; i++)
   {
      *aline++ = x[i];
      *aline++ = amp[i];
   }
   polyline2d(gfd, line, n, FALSE);
/*
   Get a new screen and draw the axes and labels.
*/
   clear_view_surface(gfd);
   wl = -50.;
   wr = 50.;
   wb = -10.;
   wt = 10.;
   xstep = 10.;
   ystep = 2.;
   ytitle = "phase (radians)";
   init_coords(gfd, wl, wb, wr, wt, .15, .15, .05, .05);
   wc_to_vdc(gfd, wl, wb, 0., &xl, &yl, &dum);
   wc_to_vdc(gfd, wr, wt, 0., &x2, &y2, &dum);
   clip_rectangle(gfd, xl, x2, yl-.001, y2);
   axis(gfd, wl, wb, xstep, ystep);
   x_labels(gfd, wl, wb, xstep, wl, xstep, wl, wr, "%2.0f.", xtitle, title);
   y_labels(gfd, wl, 0., ystep, 0., ystep, wb, wt, "%4.0f.", ytitle);
/*
   Draw the amplitude data accumulated.
*/
   aline = &line[0];
   for (i = 0 ; i <= n ; i++)
   {
      *aline++;
      *aline++ = phase[i];
   }
   polyline2d(gfd, line, n, FALSE);
/*
   Close the input file and the graphics device.
*/
   fclose(fp);
   gclose(gfd);
}
```

H.3.2 Graphics for Line Source Results

The program for graphing the line source results is more complicated than the programs for graphing the plane wave results in that there are more parameters to consider for one graph. The program vuseis thus has an input parameter file that can be used, if desired. An example of this file is presented in Section H.3.2.3 below.

The program vuseis is also written to read the modeling output data in binary, rather than ascii, form. This was done because the output files from synseis were quite large and were compacted by conversion to binary. Allowing vuseis to read the binary files saved time and effort. The program that performs the conversion is called compact, and is included in Section H.3.2.2 below.

H.3.2.1 vuseis

```
/*
   Procedure name : vuseis

   Author         : J.C. Biard

   Discussion     :
         This program produces a synthetic seismogram plot
      from the data file created by the program synseis.  Information
      is read in from an input file that specifies the plot limits and
      the scale factor to use when plotting the waveforms.
         The seismogram is a distance vs. time plot in which a number
      of waveforms are plotted on the same page.  These waveforms
      represent the signal recieved at a number of different detectors,
      which from a linear array.  Each waveform is drawn parallel to
      the time axis, with the zero amplitude level corresponding to
      the distance of the detection point from the origin.  The
      magnitude of the waveforms on the plot is arbitrary.  Only
      relative sizes of signals and positional information are intended
      to be obtained from one of these plots.  By setting the proper
      variables in the input file, it is possible to cause the signals
      above a threshold level to be blacked in, thus aiding in the
      interpretation of the data.
         The input file contains three sets of information.  The first
      set specifies the magnification scale factor for the waveforms
      in the plot and sets the threshold for the peak blacking if desired.
      The second set specifies the x dimension limits in meters.  The
      third set specifies the time axis limits for the waveforms.  If
```

no input file is included, then default limits for these values are used.

The command line arguments include a single character argument that specifies if only the "i"ncident, "s"cattered, or "b"oth incident and scattered waveforms are to be plotted in the seismogram. There is also an optional argument specifying an alternate graphics device for the plot.

Invocation : The program is invoked by the command

```
vuseis [bis] [infile] [gdevice] [gdriver] < datafile
```

where the arguments are: (bracketed items optional)

b	Plot both incident and scattered waveforms.
i	Plot only incident waveforms.
s	Plot only scattered waveforms.
infile	Input data file for vuseis.
gdevice	Device special file for the graphics device.
gdriver	Name of the starbase graphics driver for gdevice.
datafile	Output file from synseis.

The input data file has the form:

```
100., 0.          # Scale, blacking threshold (<0-left,>0-right,=0-none)
0., 0.            # left x limit, right x limit.
0., 120.          # starting time limit, stopping time limit.
```

With no arguments but the indirection of the synseis datafile, the program will plot all the waveforms in the file, both incident and scattered, from the leftmost x limit to the rightmost, from zero time to the maximum time found, with a unity scale factor and no peak blacking. The graphics output will, by default, go to the terminal from which the program was invoked, which is assumed to be compatible with an HP 26xx class graphics terminal.

Procedures called from this module :

axis	Draws a pair of x- and y-axes with optional tick marks.
init_coords	Sets up the graphics window and character sizes.
x_labels	Draws the labels on the x-axis and the plot head.
y_labels	Draws the labels on the y-axis.

Starbase graphics procedures called from this module :
(These program codes are not the property of SWRI and cannot be published in this report.)

clear_view_surface	Clears the screen or ejects the page.
clip_rectangle	Sets the software clip range.
gclose	Empties buffers and closes the graphics device.
gopen	Opens and initializes the graphics device.
interior_style	Specifies the fill attributes for polygons.
line_color_index	Specifies which pen to use in plotting.
partial_polygon2d	Builds a set of incomplete polygon edges.

```
     polygon2d           Draws a closed polygon or a series of incomplete
                            polygons.
     polyline2d          Draws a line through a number of points.
     text_color_index    Specifies the pen to use when drawing text.
     wc_to_vdc           Converts a position from the world coordinate
                            system to the virtual device coordinate
                            system.

   Formal Arguments :

     Input :
        argc        The number of command line arguments + 1.
        argv        The array of command line arguments.
           argv[0] The name of the procedure being executed.
           argv[1] (Opt.) Indicates the type of waveform to plot.
           argv[2] (Opt.) The input data filename.
           argv[3] (Opt.) The graphics device special filename.
           argv[4] (Opt.) The graphics driver.

     Output :
        none

   External variables :
     none
*/

#include <stdio.h>
#include <math.h>
#include <starbase.c.h>

main(argc, argv)
int argc;
char *argv[];
{
   char    *xtitle, *ytitle, *title2, *point;
   int     i, n, gfd, j, k, xflag, tflag, wflag, intbuf[3];
   int     nx, np, nrefl;
   float   xl, x2, y!, y2, dum, pi, extent[12];
   float   wl, wr, wb, wt, xstep, ystep;
   float   scale, thresh, xscale, xlolim, xhilim, tlolim, thilim;
   float   line[16384], *aline;
   double h, cl, rhol, c2, rho2, w0, alpha, df, dt, a, xc, yc, xs, ys, x0, y0;
   double ql, wl, q2, w2, dx;
   double tdi, tds, xd, inc[4096], scat[4096], *ap;
   FILE    *fpi;

   char    *gets();
   double sqrt();
   FILE    *fopen();

   pi = 3.141592654;
/*
   Set the plotting parameters according to the input arguments.
*/
```

```
   switch(argc)
   {
/*
   If no graph option, input parameter file or driver information is
      given at run time, set to plot all and open the terminal as the
      graphics device.
*/
      case 1:
         wflag = 2;
         scale = 1.;
         thresh = 0.;
         xlolim = xhilim = 0.;
         tlolim = thilim = 0.;
         gfd = gopen("/dev/tty", OUTDEV, "hpterm", INIT);
         break;
      case 2:
/*
   If only the graph option is given at run time, select the graph to
      plot, give the maximum ranges, and open the terminal as the graphics
      device.
*/
         switch(argv[1][0])
         {
            case 'i':
               wflag = 0;
               break;
            case 's':
               wflag = 1;
               break;
            case 'b':
               wflag = 2;
               break;
            default:
               exit(-1);
         }
         scale = 1.;
         thresh = 0.;
         xlolim = xhilim = 0.;
         tlolim = thilim = 0.;
         gfd = gopen("/dev/tty", OUTDEV, "hpterm", INIT);
         break;
      case 3:
/*
   If the graph option and input parameter files are given, set
   the appropriate values and flags and open the terminal as the
   graphics device.
*/
         switch(argv[1][0])
         {
            case 'i':
               wflag = 0;
               break;
            case 's':
               wflag = 1;
               break;
            case 'b':
               wflag = 2;
               break;
            default:
               exit(-1);
         }
/*
   Read the plotting parameters from the parameter file.
*/
         fpi = fopen(argv[2], "r");
         fscanf(fpi,"%f,%f%*[^\n]%f,%f%*[^\n]%f,%f",
                &scale, &thresh, &xlolim, &xhilim, &tlolim, &thilim);
         fclose(fpi);
         gfd = gopen("/dev/tty", OUTDEV, "hpterm", INIT);
         break;
      case 5:
/*
   If the graphics option, parameter file, and graphics device information
      is all supplied at run time, set the appropriate flags and values and
      open the specified graphics device.
*/
         switch(argv[1][0])
         {
            case 'i':
               wflag = 0;
               break;
            case 's':
               wflag = 1;
               break;
            case 'b':
               wflag = 2;
               break;
            default:
               exit(-1);
         }
/*
   Read the plotting parameters from the parameter file.
*/
         fpi = fopen(argv[2], "r");
         fscanf(fpi,"%f,%f%f,%f%f,%f",
                &scale, &thresh, &xlolim, &xhilim, &tlolim, &thilim);
         fclose(fpi);
/*
   Open the graphics device.
*/
         gfd = gopen(argv[3], OUTDEV, argv[4], INIT);
         break;
      default:
         exit(-1);
   }
```

```
/*
   If the default limits are taken set xflag FALSE, else TRUE.
*/
   if (xlolim != 0. || xhilim != 0.)
   {
      xflag = TRUE;
   }
   else
   {
      xflag = FALSE;
   }
/*
   If the default values are not taken set tflag TRUE, else FALSE.
*/
   if (tlolim != 0. || thilim != 0.)
   {
      tflag = TRUE;
   }
   else
   {
      tflag = FALSE;
   }
/*
   Read in the first 12 bytes of the input file.
      Store the values obtained in the appropriate variables.
*/
   point = (char *)intbuf;
   fread(point, sizeof(nx), 3, stdin);
   nx = intbuf[0];
   np = intbuf[1];
   nrefl = intbuf[2];
/*
   Read in the next 168 bytes of the input file.
      Store the values obtained in the appropriate variables.
*/
   point = (char *)inc;
   fread(point, sizeof(h), 21, stdin);
   h = inc[0];
   cl = inc[1];
   rhol = inc[2];
   c2 = inc[3];
   rho2 = inc[4];
   w0 = inc[5];
   alpha = inc[6];
   df = inc[7];
   dt = inc[8];
   a = inc[9];
   xc = inc[10];
   yc = inc[11];
   xs = inc[12];
   ys = inc[13];
   x0 = inc[14];
   y0 = inc[15];
   q1 = inc[16];
   wl = inc[17];
   q2 = inc[18];
   w2 = inc[19];
   dx = inc[20];
/*
   Clear the view surface and set the default pens.
*/
   clear_view_surface(gfd);
   text_color_index(gfd, 5);
   line_color_index(gfd, 1);
/*
   Set the window limits in the x (horizontal) and t (vertical) dimensions.
*/
   if (xflag == TRUE)
   {
      i = (int)(xlolim / dx);
      j = (int)(xhilim / dx);
      wl = (float)i * dx;
      wr = (float)j * dx;
      xscale = (wr - wl) / (2. * (float)(j - i));
   }
   else
   {
      wl = x0;
      wr = x0 + (float)(nx-1) * dx;
      xscale = (wr - wl) / (2. * (float)(nx - 1));
   }
   wl -= xscale;
   wr += xscale;
   if (tflag == TRUE)
   {
      wb = -thilim;
      wt = -tlolim;
   }
   else
   {
      wb = -1000. * (sqrt(4. * yc*yc + (wr-wl)*(wr-wl)) / c2 + (double)np * dt);
      wt = 0.;
   }
/*
   Set the tic mark step size in x and t.
*/
   if ((xstep = (float)((int)((wr - wl) / 10.))) == 0.)
   {
      xstep = (float)((int)((wr - wl) * 10.)) / 100.;
   }
   if ((ystep = (float)((int)((wt - wb) / 10.))) == 0.)
   {
      ystep = (float)((int)((wt - wb) * 10.)) / 100.;
   }
/*
   Set the titles, initialize the World Coordinate System window,
      and draw the axes and labels.
*/
```

```
   xtitle = "DISTANCE (m)";
   ytitle = "TIME (mSec)";
   title2 = "SH-WAVE SYNTHETIC SEISMOGRAM";
   init_coords(gfd, wl, wb, wr, wt, .15, .15, .18, .05);
   wc_to_vdc(gfd, wl, wb, 0., &xl, &yl, &dum);
   wc_to_vdc(gfd, wr, wt, 0., &x2, &y2, &dum);
   clip_rectangle(gfd, xl, x2, yl, y2);
   axis(gfd, (x0 - xstep), wt, xstep, 0.);
   axis(gfd, wl, wt, 0., ystep);
   axis(gfd, (x0 - xstep), wb+.00001, xstep, 0.);
   xl = (wr + wl) / 2.;
   x_labels(gfd, xl, wb, xstep, xl, xstep, wl, wr, "%0.0f.", xtitle, title2);
   y_labels(gfd, wl, wt, ystep, -wt, -ystep, wb, wt, "%0.0f.", ytitle);
/*
   Loop over the number of traces to plot contained in the input file.
*/
   for (i = 0 ; i < nx ; i++)
   {
/*
   Read the next 24 bytes from the input and store them in the appropriate
      variables.
*/
      point = (char *)&inc[0];
      fread(point, 1, 24, stdin);
      xd = inc[0];
      tdi = inc[1];
      tds = inc[2];
/*
   Read the next 8*np bytes to get the direct wave signal values.
*/
      point = (char *)inc;
      fread(point, sizeof(double), np, stdin);
/*
   Read the next 8*np bytes to get the scattered wave signal values.
*/
      point = (char *)scat;
      fread(point, sizeof(double), np, stdin);
/*
   If the trace is within the plot window in x, then plot it.
*/
      if (xflag == FALSE || (xd >= xlolim && xd <= xhilim))
      {
/*
   Load the plot vector array with the base offset x value and the
      t value for each point.
*/
         aline = &aline[0];
         for (n = 0 ; n < 16384 ; n += 2)
         {
            *aline++ = xd;
            *aline++ = -(float)(n / 2) * dt * 1000.;
         }
```

```
/*
   If the direct wave signal is to be calculated, add the signal values,
      with the appropriate offset in the t direction, to the plot vector
      array x values.  Scale the added values appropriately.
*/
         if (wflag == 0 || wflag == 2)
         {
            j = (int)(tdi / dt) * 2;
            if (j >= 0)
            {
               aline = &line[j];
               j = 0;
            }
            else
            {
               aline = &line[0];
               j = -j / 2;
            }
            for (n = 0 ; n < np ; n++)
            {
               *aline++ += scale * xscale * inc[n+j];
               aline++;
            }
         }
/*
   If the scattered wave signal is to be plotted, add the scattered
      wave signal x values, with the appropriate offset in t, to the
      plot vector array x values.  Scale the scattered wave signal
      appropriately.
*/
         if (wflag == 1 || wflag == 2)
         {
            j = (int)(tds / dt) * 2;
            if (j >= 0)
            {
               aline = &line[j];
               j = 0;
            }
            else
            {
               aline = &line[0];
               j = -j / 2;
            }
            for (n = 0 ; n < np ; n++)
            {
               *aline++ += scale * xscale * scat[n+j];
               aline++;
            }
         }
/*
   Draw the signal trace.
*/
         interior_style(gfd, INT_SOLID);
         polyline2d(gfd, &line[0], 8192, FALSE);
```

```
/*
   If the threshold filling option has been selected, find and fill all
      trace regions past threshold from the base x offset value.
*/
         if (thresh != 0.)
         {
            for (n = 0, k = -1 ; n < 16384 ; n += 2)
            {
/*
   If the x value of the curve over the base offset is greater than the
      threshold, start the fill process.
*/
               if ((line[n] - xd) / thresh >= 1.)
               {
/*
   If this is the first point of the region to fill, then set the
      point of the plot vector array with the point at which the
      line crosses the threshold.
*/
                  if (k < 0)
                  {
                     k = (n == 0) ? n : n - 2;
                     line[k+1] += (line[k+3] - line[k+1])
                                * (thresh + xd - line[k])
                                / (line[k+2] - line[k]);
                     line[k] = xd + thresh;
                  }
               }
/*
   If the trace line is no longer over the threshold, then fill the
      region.
*/
               else if (k >= 0)
               {
/*
   Set the point of the plot vector array which ends the fill region
      with the point at which the trace line crosses the threshold.
*/
                  line[n+1] = line[n-1] + (line[n+1] - line[n-1])
                             * (thresh + xd - line[n-2])
                             / (line[n] - line[n-2]);
                  line[n] = xd + thresh;
                  j = 1 + (n - k) / 2;
/*
   Load the appropriate part of the plot vector array into the partial
      polygon buffer.
*/
                  partial_polygon2d(gfd, &line[k], j, FALSE, TRUE);
                  k = -1;
               }
            }
/*
   Fill all regions selected previously.
*/
               polygon2d(gfd, &line[0], 0, FALSE);
               interior_style(gfd, INT_HOLLOW);
            }
         }
   }
/*
   Close the graphics device.
*/
   gclose(gfd);
}
```

H.3.2.2 compact

```
/*
   Procedure name : compact

   Author         : J.C. Biard

   Discussion     :
        This procedure takes ascii numeric output and converts it to
     binary.  This allows a large savings in memory.

   Invocation     :
        The program is invoked by the command

           compact < ifile > ofile

     where ifile is the ascii input file and ofile is the binary
     output file.
        This program may be used as a filter.

   Procedures called from this module :
     none

   External variables :
     none
*/

#include <stdio.h>

main()
{
   char   ibuffer[80], *obuffer;
   int    n;
   double x;

/*
   Read strings of ascii characters separated by white space one at
      a time until the end of the file.
*/
```

```
   while(scanf("%s", &ibuffer[0]) > 0)
   {
/*
   Seek though the string until a decimal mark or the end of string is found.
*/
      obuffer = ibuffer;
      while (*obuffer != '.' && *obuffer != '\0')
      {
         obuffer++;
      }
/*
   If the seek stopped because it was the end of the string, consider
      it to be an integer number string and read it into an integer variable.
      Then write the binary contents of the integer variable to the output.
*/
      if (*obuffer == '\0')
      {
         sscanf(&ibuffer[0], "%d", &n);
         obuffer = (char *)&n;
         fwrite(obuffer, sizeof(n), 1, stdout);
      }
/*
   If the seek stopped because a decimal mark was found, consider
      it to be a decimal number string and read it into a double variable.
      Then write the binary contents of the double variable to the output.
*/
      else if (*obuffer == '.')
      {
         sscanf(&ibuffer[0], "%lg", &x);
         obuffer = (char *)&x;
         fwrite(obuffer, sizeof(x), 1, stdout);
      }
      else
/*
   If none of the above works, write an error message.
*/
      {
         fprintf(stderr, "Unrecognizable string: %s\n", ibuffer);
      }
/*
   Push the output into the output file.
*/
      fflush(stdout);
   }
}
```

H.3.2.3 Input Parameter File for vuseis

This example input file for the program vuseis specifies the scale, threshold limit, x limits, and t limits for the graph. The form of the file must conform exactly to this example in terms of which value is on which line, although the comments are not necessary.

```
100., 0.            # Scale, blacking threshold (<0-left,>0-right,=0-none)
0., 0.              # left x limit, right x limit.
0., 120.            # starting time limit, stopping time limit.
```

H.3.3 Common Graphics Subroutines

These subroutines are used by the plane wave and line source graphing programs to perform the repetitive tasks of coordinate system initialization, character sizing, axis drawing, and labeling.

H.3.3.1 axis

```
/*
   Procedure           : axis

   Author              : J.C. Biard

   Discussion          :
         This procedure draws a pair of axes and optionally draws
      tic marks on the axes.

   Formal Arguments

      Input            :
         gfd              File descriptor of the graphics device special file.
         x0               X location of the origin of the axes.
         y0               Y location of the origin of the axes.
         xstep            X axis tic mark spacing.
         ystep            Y axis tic mark spacing.

      Output           : none

   External Variables : none
*/

axis(gfd, x0, y0, xstep, ystep)
int gfd;
float x0, y0, xstep, ystep;
{
   int cmap;
   float physlim[2][3], res[3], p1[3], p2[3];
   float vl, vr, vb, vt, wl, wr, wb, wt, dum, vtx, vty, tx, ty;
   float x, y;

/*
   Get the plotting surface limits in VDC and WC coordinate systems.
*/
   inquire_sizes(gfd, physlim, res, p1, p2, &cmap);
   vl = 0.;
   vr = (physlim[1][0] - physlim[0][0]) / (physlim[1][1] - physlim[0][1]);
   vb = 0.;
   vt = 1.;
   vtx = vty = .015;
   vdc_to_wc(gfd, vl, vb, 0., &wl, &wb, &dum);
   vdc_to_wc(gfd, vr, vt, 0., &wr, &wt, &dum);
/*
   Get the tic mark lengths in the WC.
*/
   vdc_to_wc(gfd, vtx, vty, 0., &tx, &ty, &dum);
   tx -= wl;
   ty -= wb;
/*
   Draw the axis lines.
*/
   move2d(gfd, wl, y0);
   draw2d(gfd, wr, y0);
   move2d(gfd, x0, wb);
   draw2d(gfd, x0, wt);
/*
   If xstep = 0, then don't draw x tic marks.
*/
   if (xstep != 0.)
   {
/*
   Draw tic marks on the x-axis from x0 to wr.
*/
      for (x = x0 ; x <= wr ; x += xstep)
      {
         move2d(gfd, x, (y0 + ty));
         draw2d(gfd, x, (y0 - ty));
      }
/*
   Draw tic marks on the x-axis from x0 to wl.
*/
      for (x = x0 ; x >= wl ; x -= xstep)
      {
         move2d(gfd, x, (y0 + ty));
         draw2d(gfd, x, (y0 - ty));
      }
   }
/*
   If ystep = 0, then don't draw y tic marks.
*/
   if (ystep != 0.)
   {
/*
   Draw tic marks on the y-axis from y0 to wt.
*/
      for (y = y0 ; y <= wt ; y += ystep)
      {
         move2d(gfd, (x0 + tx), y);
         draw2d(gfd, (x0 - tx), y);
      }
/*
   Draw tic marks on the y-axis from y0 to wb.
*/
```

```
      for (y = y0 ; y >= wb ; y -= ystep)
      {
         move2d(gfd, (x0 + tx), y);
         draw2d(gfd, (x0 - tx), y);
      }
   }
}
```

.3.3.2 init_coords

```
/*
   Procedure           : init_coords

   Author              : J.C. Biard

   Discussion          :
         This procedure sets the World Coordinate limits
      within the physical limits of the graphics device
      and sets default values for some graphics options.

   Formal Arguments

      Input            :
         gfd             File descriptor for the graphics device special file.
         wl              World Coordinate window left limit value.
         wb              World Coordinate window bottom limit value.
         wr              World Coordinate window right limit value.
         wt              World Coordinate window top limit value.
         fl              Fraction of plot surface left of the window.
         fb              Fraction of plot surface below the window.
         fr              Fraction of plot surface right of the window.
         ft              Fraction of plot surface above the window.

      Output           : none

   External Variables : none

*/

init_coords(gfd, wl, wb, wr, wt, fl, fb, fr, ft)
int gfd;
float wl, wr, wb, wt, fl, fr, fb, ft;
{
   int cmap;
   float physlim[2][3], res[3], p1[3], p2[3];
   float mat[3][2], vl, vr, vb, vt, sx, sy;

/*
   Find the physical limits of the graphics device.
*/
   inquire_sizes(gfd, physlim, res, p1, p2, &cmap);
   vl = 0.;
   vr = (physlim[1][0] - physlim[0][0]) / (physlim[1][1] - physlim[0][1]);
```

```
    vb = 0.;
    vt = 1.;
/*
    Initialize the transformation matrix to create the World Coordinate
        system as specified.
*/
    sx = (1. - fl - fr) * (vr - vl) / (wr - wl);
    sy = (1. - fb - ft) * (vt - vb) / (wt - wb);
    mat[0][0] = sx;
    mat[0][1] = 0.;
    mat[1][0] = 0.;
    mat[1][1] = sy;
    mat[2][0] = fl * (vr - vl) - sx * wl;
    mat[2][1] = fb * (vt - vb) - sy * wb;
/*
    Set the Virtual Device Coordinate limits and the mapping projection.
*/
    vdc_extent(gfd, vl, vb, 0., vr, vt, 0.);
    mapping_mode(gfd, TRUE);
/*
    Set the fill style for polygons to empty.
*/
    interior_style(gfd, INT_HOLLOW);
/*
    Install the World Coordinate system.
*/
    pop_matrix(gfd);
    push_vdc_matrix(gfd);
    replace_matrix2d(gfd, mat);
}
```

H.3.3.3 character_size

```
/*
    Procedure           : character_size

    Author              : J.C. Biard

    Discussion          :
            This procedure sets the character size and aspect ratio.

    Formal Arguments

        Input           :
            gfd           File descriptor for the graphics device special file.
            height        The character height in Virtual Device Coordinates.
            factor        The aspect ratio of the characters.

        Output          : none

    External Variables : none
*/
```

```
character_size(gfd, height, factor)
int gfd;
float height, factor;
{
   float wh, wh0, dum;

/*
   Obtain the height of the characters in World Coordinates.
*/
   vdc_to_wc(gfd, 0., 0., 0., &dum, &wh0, &dum);
   vdc_to_wc(gfd, 0., height, 0., &dum, &wh, &dum);
   wh -= wh0;
/*
   Set the character height.
*/
   character_height(gfd, wh);
/*
   Obtain the character width in World Coordinates.
*/
   vdc_to_wc(gfd, 0., 0., 0., &wh0, &dum, &dum);
   vdc_to_wc(gfd, (factor * height), 0., 0., &wh, &dum, &dum);
   wh -= wh0;
/*
   Set the character width.
*/
   character_width(gfd, wh);
}
```

H.3.3.4 x_labels

```
/*
   Procedure             : x_labels

   Author                : J.C. Biard

   Discussion            :
         This procedure draws labels below an x-axis line and labels
      the graph with a title.

   Formal Arguments

      Input              :
         gfd               File descriptor of the graphics device special file.
         x0                X position of the x-axis origin in World Coordinates.
         y0                Y position of the x-axis origin in World Coordinates.
         xstep             X axis tic mark increment in World Coordinates.
         xbase             X axis origin value in user units.
         xjump             X axis tic mark increment in user units.
         xl                Left bound of the x-axis in World Coordinates.
         xr                Right bound of the x-axis in World Coordinates.
         format            Format for writing tic mark labels. (As used by printf.)
         xtitle            Axis title string.
         title             Graph title string.
      Output           : none

   External Variables : none
*/

x_labels(gfd, x0, y0, xstep, xbase, xjump, xl, xr, format, xtitle, title)
char *xtitle, *format, *title;
int gfd;
float x0, y0, xstep, xbase, xjump, xl, xr;
{
   char string[20];
   int cmap;
   float physlim[2][3], res[3], p1[3], p2[3], extent[12];
   float vl, vr, vb, vt, wl, wr, wb, wt, dum;
   float x, y;

/*
   Obtain the extent of the graphics window in World Coordinates.
*/
   inquire_sizes(gfd, physlim, res, p1, p2, &cmap);
   vl = 0.;
   vr = (physlim[1][0] - physlim[0][0]) / (physlim[1][1] - physlim[0][1]);
   vb = 0.;
   vt = 1.;
   vdc_to_wc(gfd, vl, vb, 0., &wl, &wb, &dum);
   vdc_to_wc(gfd, vr, vt, 0., &wr, &wt, &dum);
/*
   Set the size of the label characters, set the text alignment, and
      set the software clipping boundary to the Virtual Device limits.
*/
   character_size(gfd, .035, .5);
   text_alignment(gfd, TA_CENTER, TA_TOP, 0., 0.);
   clip_indicator(gfd, CLIP_TO_VDC);
/*
   Label the tic marks on the right part of the x-axis in user units.
*/
   for (x = x0, dum = xbase ; x <= xr ; x += xstep, dum += xjump)
   {
      sprintf(string, format, dum);
      text2d(gfd, x, y0, string, WORLD_COORDINATE_TEXT, FALSE);
   }
/*
   Label the tic marks on the left part of the x axis in user units.
*/
   for(x = (x0-xstep), dum = (xbase-xjump) ; x >= xl ; x -= xstep, dum -= xjump)
   {
      sprintf(string, format, dum);
      text2d(gfd, x, y0, string, WORLD_COORDINATE_TEXT, FALSE);
   }
/*
   Draw the axis title.
*/
   inquire_text_extent(gfd, "A", WORLD_COORDINATE_TEXT, extent);
   x = (xr + xl ) / 2.;
   y = y0 - 2. * (extent[7] - extent[4]);
   character_size(gfd, .06, .5);
   text2d(gfd, x, y, xtitle, WORLD_COORDINATE_TEXT, FALSE);
   text_alignment(gfd, TA_CENTER, TA_TOP, 0., 0.);
/*
   Draw the graph title.
*/
   y = wt;
   character_size(gfd, .05, .5);
   text2d(gfd, x, y, title, WORLD_COORDINATE_TEXT, FALSE);
   clip_indicator(gfd, CLIP_TO_RECT);
}
```

H.3.3.5 y_labels

```
 *
   Procedure           : y_labels

   Author              : J.C. Biard

   Discussion          :
         This procedure draws labels to the left of a y-axis line.

   Formal Arguments

      Input            :
         gfd             File descriptor of the graphics device special file.
         x0              X position of the y-axis origin in World Coordinates.
         y0              Y position of the y-axis origin in World Coordinates.
         ystep           Y axis tic mark increment in World Coordinates.
         ybase           Y axis origin value in user units.
         yjump           Y axis tic mark increment in user units.
         yb              Lower bound of the y-axis in World Coordinates.
         yt              Upper bound of the y-axis in World Coordinates.
         format          Format for writing tic mark labels. (As used by printf.)
         ytitle          Axis title string.

      Output           : none

   External Variables : none
*/

y_labels(gfd, x0, y0, ystep, ybase, yjump, yb, yt, format, ytitle)
char *ytitle, *format;
int gfd;
float x0, y0, ystep, ybase, yjump, yb, yt;
{
   char string[20];
   int cmap;
   float physlim[2][3], res[3], p1[3], p2[3], extent[12];
   float vl, vr, vb, vt, wl, wr, wb, wt, dum;
   float x, y, maxlen;
```

```
/*
   Obtain the extent of the graphics window in World Coordinates.
*/
   inquire_sizes(gfd, physlim, res, p1, p2, &cmap);
   vl = 0.;
   vr = (physlim[1][0] - physlim[0][0]) / (physlim[1][1] - physlim[0][1]);
   vb = 0.;
   vt = 1.;
   maxlen = 0.;
   vdc_to_wc(gfd, vl, vb, 0., &wl, &wb, &dum);
   vdc_to_wc(gfd, vr, vt, 0., &wr, &wt, &dum);
/*
   Set the size of the label characters, set the text alignment, and
      set the software clipping boundary to the Virtual Device limits.
*/
   character_size(gfd, .035, .5);
   text_alignment(gfd, TA_RIGHT, TA_BASE, 0., 0.);
   clip_indicator(gfd, CLIP_TO_VDC);
/*
   Shift the x0 value over by the width of one character.
*/
   inquire_text_extent(gfd, "A", WORLD_COORDINATE_TEXT, extent);
   x0 -= .5 * (extent[9] - extent[3]);
/*
   Label the tic marks on the upper part of the y-axis in user units.
*/
   for (y = y0, dum = ybase ; y <= yt ; y += ystep, dum += yjump)
   {
      sprintf(string, format, dum);
      text2d(gfd, x0, y, string, WORLD_COORDINATE_TEXT, FALSE);
      inquire_text_extent(gfd, string, WORLD_COORDINATE_TEXT, extent);
      x = extent[9] - extent[3];
      maxlen = (x > maxlen) ? x : maxlen;
   }
/*
   Label the tic marks on the lower part of the y-axis in user units.
*/
   for(y = (y0-ystep), dum = (ybase-yjump) ; y >= yb ; y -= ystep, dum -= yjump)
   {
      sprintf(string, format, dum);
      text2d(gfd, x0, y, string, WORLD_COORDINATE_TEXT, FALSE);
      inquire_text_extent(gfd, string, WORLD_COORDINATE_TEXT, extent);
      x = extent[9] - extent[3];
      maxlen = (x > maxlen) ? x : maxlen;
   }
/*
   Draw the axis title.
*/
   character_size(gfd, .06, .5);
   text_alignment(gfd, TA_RIGHT, TA_HALF, 0., 0.);
   inquire_text_extent(gfd, "A", WORLD_COORDINATE_TEXT, extent);
   y = (yb + yt ) / 2.;
   x = x0 - 2.5 * (extent[9] - extent[3]) - maxlen;
   text2d(gfd, x, y, ytitle, WORLD_COORDINATE_TEXT, FALSE);
```

```
    text_alignment(gfd, TA_RIGHT, TA_CAP, 0., 0.);
    x += extent[9] - extent[3];
    text2d(gfd, x, y, "s", WORLD_COORDINATE_TEXT, FALSE);
    clip_indicator(gfd, CLIP_TO_RECT);
}
```

H.3.3.6 y_labels_r

```
/*
    Procedure          : y_labels_r

    Author             : J.C. Biard

    Discussion         :
        This procedure draws labels to the right of a y-axis line.

    Formal Arguments

       Input           :
          gfd            File descriptor of the graphics device special file.
          x0             X position of the y-axis origin in World Coordinates.
          y0             Y position of the y-axis origin in World Coordinates.
          ystep          Y axis tic mark increment in World Coordinates.
          ybase          Y axis origin value in user units.
          yjump          Y axis tic mark increment in user units.
          yb             Lower bound of the y-axis in World Coordinates.
          yt             Upper bound of the y-axis in World Coordinates.
          format         Format for writing tic mark labels. (As used by printf.)
          ytitle         Axis title string.

       Output          : none

    External Variables : none
*/

y_labels_r(gfd, x0, y0, ystep, ybase, yjump, yb, yt, format, ytitle)
char *ytitle, *format;
int gfd;
float x0, y0, ystep, ybase, yjump, yb, yt;
{
    char string[20];
    int cmap;
    float physlim[2][3], res[3], p1[3], p2[3], extent[12];
    float vl, vr, vb, vt, wl, wr, wb, wt, dum;
    float x, y, maxlen;

/*
    Obtain the extent of the graphics window in World Coordinates.
*/
    inquire_sizes(gfd, physlim, res, p1, p2, &cmap);
    vl = 0.;
    vr = (physlim[1][0] - physlim[0][0]) / (physlim[1][1] - physlim[0][1]);
    vb = 0.;
    vt = 1.;
    maxlen = 0.;
    vdc_to_wc(gfd, vl, vb, 0., &wl, &wb, &dum);
    vdc_to_wc(gfd, vr, vt, 0., &wr, &wt, &dum);
/*
    Set the size of the label characters, set the text alignment, and
       set the software clipping boundary to the Virtual Device limits.
*/
    character_size(gfd, .04, .5);
    text_alignment(gfd, TA_LEFT, TA_HALF, 0., 0.);
    clip_indicator(gfd, CLIP_TO_VDC);
/*
    Shift the x0 value over by the width of one character.
*/
    inquire_text_extent(gfd, "A", WORLD_COORDINATE_TEXT, extent);
    x0 -= .5 * (extent[9] - extent[3]);
/*
    Label the tic marks on the upper part of the y-axis in user units.
*/
    for (y = y0, dum = ybase ; y <= yt ; y += ystep, dum += yjump)
    {
       sprintf(string, format, dum);
       text2d(gfd, x0, y, string, WORLD_COORDINATE_TEXT, FALSE);
       inquire_text_extent(gfd, string, WORLD_COORDINATE_TEXT, extent);
       x = extent[9] - extent[3];
       maxlen = (x > maxlen) ? x : maxlen;
    }
/*
    Label the tic marks on the lower part of the y-axis in user units.
*/
    for(y = (y0-ystep), dum = (ybase-yjump) ; y >= yb ; y -= ystep, dum -= yjump)
    {
       sprintf(string, format, dum);
       text2d(gfd, x0, y, string, WORLD_COORDINATE_TEXT, FALSE);
       inquire_text_extent(gfd, string, WORLD_COORDINATE_TEXT, extent);
       x = extent[9] - extent[3];
       maxlen = (x > maxlen) ? x : maxlen;
    }
/*
    Draw the axis title.
*/
    character_size(gfd, .04, .5);
    text_alignment(gfd, TA_LEFT, TA_HALF, 0., 0.);
    inquire_text_extent(gfd, "A", WORLD_COORDINATE_TEXT, extent);
    y = (yb + yt ) / 2.;
    x = x0 + 2.5 * (extent[9] - extent[3]) + maxlen;
    text_path(gfd, PATH_DOWN);
    text2d(gfd, x, y, ytitle, WORLD_COORDINATE_TEXT, FALSE);
    text_path(gfd, PATH_RIGHT);
    clip_indicator(gfd, CLIP_TO_RECT);
}
```

LIST OF REFERENCES

VOLUME II

Aki, K. and Richards, P.G., Quantitative Seismology, Theory and Methods, W.H. Freeman and Company, New York, 1980.

deHoop, A.T., "Modification of Cagniard's Method for Solving Seismic Pulse Problems," Applied Science Research, B8, pp. 349-356, 1960.

deHoop, A.T., "Pulsed Electromagnetic Radiation from a Line Source in a Two-Media Configuration," Radio Science, Vol. 14, No. 2, pp. 253-268, 1979.

El-Akily and Datta, S.K., "Response of a Cylindrical Shell to Disturbances in a Half-Space, Earthquake Engineering and Structural Dynamics, Vol. 8, pp. 469-477, 1980.

Gradshteyn, I.S. and Ryzhik, I.M., Table of Integrals, Series, and Products, Academic Press, N.Y., p. 1086, 1965.

Kamel, A. and Felsen, L.B., "Hybrid Ray-Mode Formulation of SH Motion in a Two-Layer Half-Space," Bulletin of the Seismological Society of America, Vol. 71, No. 6, pp. 1763-1781, 1981.

Kong, J.A., Electromagnetic Wave Theory, John Wiley and Sons, Inc., New York, 1986.

Stinson, D.C., Intermediate Mathematics of Electromagnetics, Prentice-Hall, Inc., New Jersey, 1976.